580213-5 Introduktion till datateknik

Lindén, Greger

University of Helsinki, Department of Communication

1998

Lindén , G 1998 , 580213-5 Introduktion till datateknik . Series of Publications D , nr. D395 , University of Helsinki, Department of Communication . < http://www.cs.helsinki.fi/u/linden/teaching/data98/Introduktion_till_datateknik.pdf >

http://hdl.handle.net/10138/17723

acceptedVersion

580213-5 Introduktion till datateknik

Föreläsningsunderlag hösten 1998
Institutionen för datavetenskap
Greger Lindén
Helsingfors 2 september 1998

Contents

Chapter 1

Introduktion - Vad är datateknik?

1.1 Datateknik och databehandling

Databehandling eller **informationsbehandling** innebär ett utförande av en systematisk serie av operationer på givna data. Möjliga operationer kan vara t.ex. att läsa, skriva, räkna, jämföra, sortera eller minnas. Databehandling är alltså ingenting nytt som man uppfunnit i och med ”datoråldern”.

Exempel: manuell databehandling, automatisk databehandling

En **dator** eller en **datamaskin** är en apparat för automatisk databehandling. Datorn kan utföra omfattande beräkningar med ett stort antal aritmetiska operationer eller logikoperationer.

Automatisk databehandling (ADB) är databehandling som huvudsakligen utförs med datorer.

Automation innebär utnyttjande av automatiska hjälpmedel för att förverkliga en process, t. ex. i hushållsmaskiner.

Datateknik inbegriper både ADB och automation, dvs. apparater, metoder och kunskap om användning.

1.2 Datavetenskap

Datavetenskap, **databehandlingslära** eller **datalogi** (fi. tietojenkäsittelytiede, eng. computer science, ty. Informatik, fr. informatique) studerar ADB ur en vetenskaplig synvinkel.

I litteraturen förekommer olika defintioner på datavetenskap:

- **datalogi** - en vetenskapsgren som avser metoder och teknik i samband med automatisk databehandling [Dataordboken]
- **tietojenkäsittelytiede** - Tiedon esityksen ja käsittelyn periaatteet ja välineistö, tietosysteemit, ohjelmoinnin ja ohjelmistojen tekniikka sekä tietotekniikan eri aloille soveltamisen menetelmät tieteellisen tarkastelun kohteena. [ATK-sanakirja]

  (Principer och medel för datarepresentation och databehandling, datasystem, program- och programvaruteknik, tillämpning av datateknik på olika områden betraktat på ett vetenskapligt sätt.)
- **computer science** - datavetenskap granskar de algoritmiska processer med vilka information kan beskrivas och bearbetas: deras teori, analys, planering, effektivitet, förverkligande och tillämpningar. Datavetenskapens grundfråga är:

  Vad kan automatiseras (effektivt)?

  [Amerikansk rapport/studiehandboken]
- **computer science** - Computer science is the discipline that seeks to build a scientific foundation for variety of topics, including computer design, computer programming, information processing, algorithmic solutions of problems, and the algorithmic process itself [Brookshear: Computer Science: An Overview].

Datavetenskap är mycket mera än bara programmering!

1.3 Algoritmer

Ett centralt begrepp inom datavetenskapen är **algoritmen**. En algoritm innehåller väldefinierade regler för lösning av ett problem eller utförande av en uppgift i ett ändligt antal steg. Exempel på algoritmer: matrecept, instruktion för programmering av video, löneberäkningsprogram, beskrivning på hur man bygger ett modellflygplan, bruksanvisning för tvättmaskin, eller beskrivning på hur man gör trollkonster.

Algoritmer representeras som **program** i datorer. Man talar om **programvara** till skillnad från **maskinvara**. Datorn kan bara lösa en uppgift om det finns en algoritm som löser upgiften.

I matematiken har man studerat algoritmer länge.

När man funnit algoritmen, behöver man inte längre förstå varför den fungerar. Intelligensen är så att säga kodad i algoritmen.

Datorer kan bara vara så intelligenta som deras program. Om det inte finns någon algoritm för ett problem kan inte heller datorn lösa det.

Ett exempel på en algoritm: Euklides algoritm som finner den största gemensamma delaren till två tal:

1. Tilldela **m** och **n** de två talen.
2. Dela **m** med **n** och kalla resten **r**
3. Om **r** är olika 0, ge **m** värdet **n**, och **n** värdet **r** och återgå till steg 2; i övriga fall är den största gemensamma delaren det värde som finns i **n**.

Obs. Det finns problem som inte har någon känd lösning (eller algoritm).

Algoritmer behandlas även senare i avsnittet om programspråk och programvara.

1.4 Utveckling av algoritmiska maskiner

I början av 1900-talet konstruerades många teoretiska maskiner. Dessa ligger till grund för dagens datorer.

Längre bak i tiden har vi **kulramen** och **räknestickan** som två "räknemaskiner".

Pascal och Leibniz konstruerade enklare mekaniska räknemaskiner på 1600-talet.

I slutet av 1800-talet planerade engelsmannen Charles Babbage en "riktig" datamaskin men med den tidens teknik gick den inte att konstruera.

Först på 1940-talet konstruerades de första riktiga (elektroniska) datamaskinerna, stora som hus. En enkel räknedosa idag har mycket större kapacitet.

1.5 Uppkomsten av datavetenskapen

Algoritmerna var till en början begränsade av litet datautrymme och långsamma maskiner.

Man behövde forskning i datateknik och -vetenskap som kunde få fram bättre maskiner men framför allt bättre program.

Datavetenskapen försöker bl.a. besvara följande frågor:

- Vilka problem kan lösas med en algoritmisk process?
- Hur kan man lättare upptäcka algoritmer?
- Hur kan vi förbättra de tekniker med vilka vi representerar algoritmer?
- Hur kan vår kunskap om algoritmer och teknologi skapa bättre algoritmiska maskiner?
- Hur kan man jämföra egenskaper hos olika algoritmer?

Allt detta leder till studium av algoritmer.

1.6 Etiska, sociala och legala frågor

Den enorma utvecklingen leder till att lagstiftning, mm. inte hinner med.

Vem äger ett program? Vem får sälja eller ansöka om licens?

Bör datateknik och -användning begränsas av myndigheter?

Inga självklara svar finns.

Databrott är vanligt och det lönar sig?

1.7 Varför studera datavetenskap och -teknik?

Användbart - effektivt - nödvändigt

- hjälper i studierna
- effektivt arbete
- bra för svåra problem
- mera information
- underlättar kontakter
- allmänbildning
- kan inte undgås
- datorer i arbetslivet

- underhållning

Läsa - skriva - räkna - data = grundläggande färdigheter

Finland innehar en tätposition när det gäller användande av och kunnande om datorer och datorsystem.

Det finns ingen återvändo när det gäller datorerna?

Chapter 2

Data och datarepresentation

På denna föreläsning behandlar vi representation och lagring av data.

2.1 Data och information

Med **data** menar vi representation av uppgifter - fakta, begrepp eller instruktioner.

Data kan vara tal, text, teckningar, fotografier, rörliga bilder, ljud, ljus, mm.

Information är den innebörd vi lägger i data. Genom **tolkning** av data ger vi dem olika innebörd.

(Observera att inom andra vetenskaper (än datavetenskapen) kan termerna t.o.m. ha motsatt betydelse.)

Exempel:

Datorer behandlar enbart data.

Data kan representeras på många olika sätt. En klassifikation bygger på om data kan varierra kontinuerligt eller endast anta vissa värden. I det förra fallet talar vi om **analoga data**, i det senare om **diskreta data**. Om den diskreta representation enbart uttrycks med siffror (samt specialtecken) är den **digital.**

Data	Information/tolkning
240997	datum, telefonnummer, antal invånare på Island
Lovisa	stad, adress, namn
A+	blodgrupp, vitsord, kreditvärdighet

Table 2.1: Data med olika informationsinnehåll

Representation	Slag av data	Medium
Analoga data	temperaturangivelse	sprittermomemter
	kursangivelse	kompass
Diskreta data	nödrop	Morse-signal
	roman	tryckt text
Digitala data	kurs i grader och minuter	siffror, gradtecken
	datum	siffror, möjligen punkt, bindestreck eller snedstreck

Table 2.2: Olika slag av data

Datorn behandlar data digitalt.

Exempel:

2.2 Talsystem

Det vanliga talsystemet har basen 10 och är ett s.k. **positionssystem**.

Talet 4711 kan skrivas

$$4 * 1000 + 7 * 100 + 1 * 10 + 1 = 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 1 * 10^0$$

Men i datorn ...

Talsystemet med basen 2 kallas det **binära talsystemet**. Talen i det binära talsystemet brukar betecknas med nollor och ettor. En **binär siffra** (0 eller 1) kallas en **bit**, pl. bitar.

Datorn lagrar information som sekvenser (följder) av bitar. En bit representerar två lägen, av och till, vilket är lätt att representera i en dator. Nollan representerar det ena läget, ettan det andra.

(En bit (pl. bit) är också en måttenhet för minneskapacitet.)

Man grupperar bitar i **ord**. Ett ord med 8 bitar brukar kallas **byte**.

Decimalsystemet har basen tio.

Oktalsystemet har basen 8.

Hexadecimalsystemet har basen 16.

Genom **konvertering** kan man ändra tal i ett system till tal i ett annat, exempelvis konvertera binära tal till oktaltal.

Exempel

$$34_8 = 3*8^1+4*8^0 = 28_{10} = 1*16^1+12*16^0 = 1C_{16} = 1*2^4+1*2^3+1*2^2 = 11100_2$$

$$10101_2 = 1*2^4+1*2^2+1*2^0 = 16+4+1 = 21_{10}$$

Det är relativt lätt att konvertera mellan binära, oktala och hexadecimala tal. Man grupperar det binära talet i 3 eller 4 bitars grupper och kan sedan lätt konvertera det till oktal- eller hexadecimalsystemet.

Exempel

$$100001_2 = 41_8$$
$$100001_2 = 21_{16}$$

Man utför räkneoperationer i annan bas som med decimaltal.

Exempel:

$$5+15 = 20$$

$$101_2+1111_2 = 10100_2$$

$$5_8+17_8 = 24_8$$

2.3 Representation av text

För att reperesentera text i datorn använder man sig av **teckenkoder**. Man kommer överens om att en viss följd av bitar motsvarar **tecknet** A, en annan följd tecknet B, osv.

Med en bit kan man representera två olika tecken (1 och 0), med följder av två bitar kan man representera fyra olika tecken, osv. Allmänt gäller att med **n** bitar kan man representera 2^n olika tecken.

bas 10	bas 2	bas 8	bas 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

Table 2.3: Tal i olika talsystem.

Figure 2.1: Exempel på talkonvertering.

Standardiserade kodsystem

ASCII (American Standard Code for Information Interchange)

- ett tecken representeras med 7 bitar (eller utvidgat med 8)
- möjliga tecken: $2^7 = 128$ eller $2^8 = 256$

Teckensträngen Hello kodas då i ASCII som

1001000 1100101 1101100 1101100 1101111

En teckensträng är alltså en följd av tecken som i sin tur består av en följd av bitar.

ISO Latin1

- ett tecken representeras med 8 bitar
- möjligheter: $2^8 = 256$

Unicode

- ett tecken representeras med 16 bitar
- möjligheter: $2^{16} = 65356$

Förslag till koder med 32 bitar/tecken finns (ISO - International Standards Organization).

2.4 Principer för representation av tal

Tal kan representeras med en teckenkod, t.ex. ASCII

Exempel:

För att representera talet 100 behövs tre tecken och 21 bitar (ASCII).

Vi får att 100 motsvaras av 0110001 0110000 0110000.

Ett effektivare sätt är att representera tal binärt

Tecken	Binärt	Dec	Tecken	Binärt	Dec	Tecken	Binärt	Dec	Tecken	Binärt	Dec
NUL	0000000	0	Blank	0100000	32	@	1000000	64	‘	1100000	96
SOH	0000001	1	!	0100001	33	A	1000001	65	a	1100001	97
STX	0000010	2	"	0100010	34	B	1000010	66	b	1100010	98
ETX	0000011	3	#	0100011	35	C	1000011	67	c	1100011	99
EOT	0000100	4	$	0100100	36	D	1000100	68	d	1100100	100
ENQ	0000101	5	%	0100101	37	E	1000101	69	e	1100101	101
ACK	0000110	6	&	0100110	38	F	1000110	70	f	1100110	102
BEL	0000111	7	’	0100111	39	G	1000111	71	g	1100111	103
BS	0001000	8	(	0101000	40	H	1001000	72	h	1101000	104
HT	0001001	9	)	0101001	41	I	1001001	73	i	1101001	105
LF	0001010	10	*	0101010	42	J	1001010	74	j	1101010	106
VT	0001011	11	+	0101011	43	K	1001011	75	k	1101011	107
FF	0001100	12	,	0101100	44	L	1001100	76	l	1101100	108
CR	0001101	13	-	0101101	45	M	1001101	77	m	1101101	109
SO	0001110	14	.	0101110	46	N	1001110	78	n	1101110	110
SI	0001111	15	/	0101111	47	O	1001111	79	o	1101111	111
DLE	0010000	16	0	0110000	48	P	1010000	80	p	1110000	112
DCI	0010001	17	1	0110001	49	Q	1010001	81	q	1110001	113
DC2	0010010	18	2	0110010	50	R	1010010	82	r	1110010	114
DC3	0010011	19	3	0110011	51	S	1010011	83	s	1110011	115
DC4	0010100	20	4	0110100	52	T	1010100	84	t	1110100	116
NAK	0010101	21	5	0110101	53	U	1010101	85	u	1110101	117
SYN	0010110	22	6	0110110	54	V	1010110	86	v	1110110	118
ETB	0010111	23	7	0110111	55	W	1010111	87	w	1110111	119
CAN	0011000	24	8	0111000	56	X	1011000	88	x	1111000	120
EM	0011001	25	9	0111001	57	Y	1011001	89	y	1111001	121
SUB	0011010	26	:	0111010	58	Z	1011010	90	z	1111010	122
ESC	0011011	27	;	0111011	59	[	1011011	91	{	1111011	123
FS	0011100	28	<	0111100	60	\	1011100	92	|	1111100	124
GS	0011101	29	=	0111101	61	]	1011101	93	}	1111101	125
RS	0011110	30	>	0111110	62	^	1011110	94	~	1111110	126
US	0011111	31	?	0111111	63	_	1011111	95	DEL	1111111	127

Table 2.4: ASCII-tabellen

0.	NUL	52. 4	104. h	156.		208. Ð	
1.	SOH	53. 5	105. i	157.		209. Ñ	
2.	STX	54. 6	106. j	158.		210. Ò	
3.	ETX	55. 7	107. k	159.		211. Ó	
4.	EOT	56. 8	108. l	160.		212. Ô	
5.	ENQ	57. 9	109. m	161.	¡	213. Õ	
6.	ACK	58. :	110. n	162.	?	214. Ö	
7.	BEL	59. ;	111. o	163.	£	215. ×	
8.	BS	60. <	112. p	164.	?	216. Ø	
9.	TAB	61. =	113. q	165.	?	217. Ù	
10.	LF	62. >	114. r	166.	|	218. Ú	
11.	VT	63. ?	115. s	167.	§	219. Û	
12.	FF	64. @	116. t	168.	¨	220. Ü	
13.	CR	65. A	117. u	169.	©	221. Ý	
14.	SO	66. B	118. v	170.	?	222. Þ	
15.	SI	67. C	119. w	171.	<<	223. ß	
16.	DLE	68. D	120. x	172.	?	224. à	
17.	DC1	69. E	121. y	173.	?	225. á	
18.	DC2	70. F	122. z	174.	?	226. â	
19.	DC3	71. G	123. {	175.	?	227. ã	
20.	DC4	72. H	124. |	176.	?	228. ä	
21.	NAK	73. I	125. }	177.	±	229. å	
22.	SYN	74. J	126. ~	178.	^{2}	230. æ	
23.	ETB	75. K	127.	179.	^{3}	231. ç	
24.	CAN	76. L	128.	180.	´	232. è	
25.	EM	77. M	129.	181.	?	233. é	
26.	SUB	78. N	130.	182.	¶	234. ê	
27.	ESC	79. O	131.	183.	·	235. ë	
28.	FS	80. P	132.	184.	¸	236. ì	
29.	GS	81. Q	133.	185.	^{1}	237. í	
30.	RS	82. R	134.	186.	?	238. î	
31.	US	83. S	135.	187.	>>	239. ï	
32.	spc	84. T	136.	188.	$\frac{1}{4}$	240. ∂	
33.	!	85. U	137.	189.	$\frac{1}{2}$	241. ñ	
34.	"	86. V	138.	190.	$\frac{3}{4}$	242. ò	
35.	#	87. W	139.	191.	¿	243. ó	
36.	$	88. X	140.	192.	À	244. ô	
37.	%	89. Y	141.	193.	Á	245. õ	
38.	&	90. Z	142.	194.	Â	246. ö	
39.	’	91. [	143.	195.	Ã	247. ÷	
40.	(	92. \	144.	196.	Ä	248. ø	
41.	)	93.]	145.	197.	Å	249. ù	
42.	*	94. ^	146.	198.	Æ	250. ú	
43.	+	95. _	147.	199.	Ç	251. û	
44.	,	96. ‘	148.	200.	È	252. ü	
45.	-	97. a	149.	201.	É	253. ý	
46.	.	98. b	150.	202.	Ê	254. þ	
47.	/	99. c	151.	203.	Ë	255. ÿ	
48.	0	100. d	152.	204.	Ì		
49.	1	101. e	153.	205.	Í		
50.	2	102. f	154.	206.	Î		
51.	3	103. g	155.	207.	Ï		

Table 2.5: ISO Latin1-tabellen

Figure 2.2: Strunta i å-ä-ö.

Figure 2.3: Strunta i å-ä-ö, nr 2.

Exempel

$$100_{10} = 1100100_2$$

För talet 100 behövs det bara 7 bitar.

Med **n** bitar kan man således representera talen $0..2^n - 1$.

Förtecken - belopp

Heltal kan representeras med en bit för förtecken (positivt eller negativt) och övriga bitar för talets absolutbelopp.

Obs. Det finns i så fall två nollor.

Exempel: Anta att vi kan använda 8 bitar. Då har vi t.ex. (förtecknet representeras av den första biten):

$$+105_{10} = 01101001$$

$$-105_{10} = 11101001$$

$$+0_{10} = 00000000$$

$$-0_{10} = 10000000$$

Men aritmetiken blir besvärlig

1-komplementmetoden

All bitar komplementeras (0 blir 1 och 1 blir 0) för negativa tal. Alla positiva tal måste inledas med en nolla.

Exempel

$$+105_{10} = 01101001$$

$$-105_{10} = 10010110$$

Det blir lättare att utföra aritmetik (överkurs).

2-komplement-metoden

Som 1-komplementmetoden. Dessutom adderas en etta till negativa tal.

Exempel

$$+105_{10} = 01101001$$

$$-105_{10} = 10010111$$

även här är aritmetiken naturlig (överkurs).

Excess-n-metoden

Excess-n-metoden förskjuter hela talområdet genom addition av en konstant **n**. Om k bitar används blir $n = 2^{k-1}$, exempelvis $k = 8$ och $n = 128$.

Exempel

Talet -105 lagras som $128 - 105 = 23$ och talet $+105$ som talet $128 + 105 = 233$

$$-105_{10} = 00010111$$

$$+105_{10} = 11101001$$

Det minsta talet som kan lagras (-128) lagras alltså som 00000000, följande (-127) som 00000001, osv.

2.5 Talrepresentation

Heltal

Bitföljden tolkas som om binärpunkten fanns genast till höger.

Bråktal

Man kan lagra bråktal i intervallet $[-1, +1[$.

Exempel

Anta att 2-komplementmetoden används. Ett ord tolkas som om det fanns en binärpunkt mellan första och andra biten.

$00110000 = 0 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3} = 3/8$

Flyttal

Man representerar tal som en produkt av två tal, **mantissa** och **karakteristika**, t.ex. i formen $0.317 * 10^3 = 317$.

Exempel

signum	karakteristika	mantissa
0	0100010	$10011110100000000000000_2 =$

2.6 Representation av logiska värden

Logiska värden kan representeras med bara en bit (sant eller falskt).

2.7 Representation av ljud

Ljud lagras som en följd av heltal som representerar samplingsvärden. Ju bättre ljud desto mer bitar behövs. CD-kvalitet motsvarar 44 100 st. 16-bitars värden spelas upp per sekund.

Analogt ljud måste först ändras om till digitalt

2.8 Representation av bilder

I huvudsak två lagringsmöjligheter.

Raster- eller pixelgrafik: Varje bildpunkt lagras med 1 till 32 bitar. Resultatet blir en **bitkarta**. TIFF (Tag Image File Format), GIF (Graphic Interchange Format) och JPEG (Joint Photographic Experts Group) är exempel på representationssätt för bitkartor.

Vektorgrafik: Man beskriver de objekt som finns med i bilden, t.ex. för en linje ger man begynnelse- och slutkoordinater samt tjocklek och färg.

Rastergrafik:

Färgfotografier kan kräva flera megabyte lagringsutrymme.

Ju fler bildpunkter desto bättre **resolution**.

Bildskärmen på en mikrodator kan innehålla 640 * 350 punkter och varje punkt representeras av 4 bitar. Då krävs det 896 000 bitar för att representera hela bildskärmen.

Arbetsstationer kan ha mycket högre resolution. Färger kräver dessutom fler bitar per punkt.

Rörliga bilder kräver snabb uppdatering av bildskärmen. Ett representationssätt kallas MPFG (Motion Picture Experts Group).

Ofta representerar man bara skillnaden mellan två på varandra följande bilder och sparar på så sätt utrymme.

2.9 Redundanskontroller

Vid dataöverföring finns det risk för att bitar förvrängs, dvs. byter värde. För att bekräfta att data är korrekta kan man lägga till redundant information (överflödig, som inte tillför något) som fungerar som kontrollinformation.

Paritetskontroll

Koden för ett tecken (t.ex. 7-bitars ASCII) komplementeras med en extra bit så att antalet ettor alltid är udda (**udda paritet**). Då vet datorn att ett tecken innehåller fel om det finns ett jämnt antal ettor i tecknet och kan be om detta tecken på nytt.

Exempel

Tecknet A motsvaras av 1000001. Med en paritetsbit (udda paritet) får vi 11000001.

Pariteten kan också vara jämn. Då lägger vi till en bit så att antalet ettor alltid är jämnt.

Men vad händer om det blir två fel i ett tecken?

Felkorrigerande koder

Tecken kan också kodas så att datorn själv kan lista ut hur tecknet skall korrigeras (överkurs, se t.ex. Brookshear, s. 50)

Chapter 3

Datorns uppbyggnad

3.1 Datorns funktionsprinciper

Några allmänna principer för en dators uppbyggnad:

- Reglerna för hur en uppgift skall lösas samt det utgångsdata som behövs finns lagrade i primärminnet
- Processorn tar fram en regel åt gången från minnet och utför de åtgärder som anges i regeln
- Om regeln anger att datat lagrat i minnet skall behandlas tar processorn fram det data som behövs och lagrar resultaten i minnet
- Kommunikationen med omvärlden sköts via in- och utmatningsenheter
- Funktionsreglerna uppstår genom att man skriver in ett program i datorns minne via någon inmatningsenhet
- Principen med program lagrade i minne (”von Neumanns princip”) är från 1940-talet, men datorerna fungerar fortfarande enligt denna princip

I användarprogrammet finns t.ex. en instruktion att multiplicera två tal. Instruktionerna som berättar hur t.ex. en multiplikation skall utföras finns i **operativsystemet**.

Operativsystemet innehåller grundinstruktionerna på vilka alla andra instruktioner baserar sig.

I **primärminnet** lagras operativsystemet, användarprogrammen och de data som dessa program behöver.

Minnet är indelat i **minnesceller**, var och en med en egen adress. En minnescell kan rymma t.ex. 4 bokstäver.

Styrenheten håller reda på vad som skall göras. I **aritmetikenheten** utförs t.ex. addition.

3.2 Datorns delar

En konventionell dator består av tre delar: en **styrenhet** en **aritmetikenhet** och ett **primärminne**.

Styrenheten och aritmetikenheten tillsammans bildar **processenheten** eller **processorn**; om man kopplar dem till ett primärminne har man en **centralenhet**).

Data flödar mellan alla dessa tre enheter; styrsignaler går dock endast från styrenheten till de två övriga.

I primärminnet lagras både program och data.

Program innehåller **instruktioner** som är avsedda att tolkas av styrenheten, medan **data** är avsedda att bearbetas (läsas, skrivas, beräknas) av aritmetikenheten.

Datorns **prestanda**: hur fort program utförs (= exekveras). Beror på **klockfrekvens** (anges i Megaherz) och instruktioner. Exempel: 33 Mz, 75 MHz, 90 MHz, 140 MHz.

En viss instruktion kräver ett visst antal **klockcykler**. Man mäter (medel-)antalet instruktioner en dator kan utföra i **Mips** (miljoner instruktioner per sekund). I persondatorer är det normalt med 1-10 Mips.

3.3 Primärminne

I primärminnet lagras de program och data som behövs (för tillfället). Kallas också **RAM** (Random access memory). Då datorn stängs försvinner de data som finns i primärminnet. Därför behövs sekundärminne.

I **sekundärminnet** lagras övriga program(-delar) och data. Sekundärminne kan t.ex. vara disketter, hårdskivor, magnetband, mm. (Se senare avsnitt.)

Primärminnet är dyrare och begränsat, sekundärminne finns att tillgå.

Primärminnet är direkt åtkomligt för processorn. Kallas därför också **inre** eller

Figure 3.1: Exempel på datorns struktur.

internt minne.

Sekundärminnet är åtkomligt endast via primärminnet.

Man talar också om **arbetsminne** (primärminne) respektive **massminne** (sekundärminne).

Primärminnet består av **celler** med unika **adresser**.

Varje cell innehåller ett **ord**. Med hjälp av adressen hittar man celler i primärminnet och kan avläsa ordets värde.

Antalet bitar i ett ord kallas **ordlängd**. Ordlängden varierar mellan olika datorer, exempelvis 8, 16 eller 32 är vanliga ordlängder.

En **minnesaccess** betyder att ett ord hämtas till processorn. Det tar ungefär 100ns -1μs att göra en minnseaccess.

Ett ord med längden 8 kallas **byte** eller **bitgrupp**. Minneskapacitet brukar mätas i byte (B), kilobyte (kB), megabyte (MB) eller gigabyte (GB).

I datorvärlden är 1 kB = 1024 byte och 1 MB = 1024 kB = 1 048 576 B och 1 GB = 1024 MB = 1 048 576 kB = 1 073 741 824 B. Men man använder också 1 kB = 1000 byte osv.

Primärminnets storlek varierar i olika datorer. Ofta kan man expandera primärminnet genom att köpa tilläggsminne.

3.4 Fickminne och virtuellt minne

Fickminnet är ett snabbt (litet) minne för temporär lagring av data under behandling. Logiskt sett ligger det mellan processorn och primärminnet.

Virtuellt minne är en teknik där datorn luras att tro att primärminnet är större än det är. Program som utförs får referera till ett större adressutrymme än vad primärminnet skulle ge tillåtelse till. Vid behov hämtas de delar som fattas från sekundärminnet (och samtidigt får vissa andra delar då strykas i primärminnet).

3.5 Processorn

Processorn består alltså av styrenheten och en aritmetikenhet samt en uppsättning **register**.

Exempel på processoroer

Intel 80486 Intel Pentium Motorola 68040 PowerPC 601

Styrenhet

Styrenheten har till uppgift att hämta en instruktion i taget från primärminnet och utföra (exekvera) den.

Som hjälp finns ett antal **register** (minnesplatser) i styrenheten där processorn kan lagra mellanresultat. Exempelvis **programräknaren** (PR) håller adressen till den instruktion som just utförs, medan instruktionen finns i **instruktionsregistret** (IR). Allmänna register kan användas för lagring av godtyckliga data till skillnad från specialregistren ovan.

De instruktioner som styrenheten kan använda sig av kallas datorns **instruktionsmängd** och utgör **maskinspråket**.

En maskininstruktion består av en **operationsdel** och en **adressdel**. Operationsdelen berättar vad som skall utföras och adressdelen visar på den minnescell som är objektet för utförandet, t.ex. lagra ett ord i en viss cell eller läs ett ord från en viss cell.

Adressdelen kan ibland ha flera adresser. Instruktionerna blir då effektivare men kräver också mera utrymme (längre ord).

Utförandet sker vanligtvis sekventiellt; processorn läser en instruktion i taget i ordningsföljd. En instruktion kan också säga åt processorn att hoppa över ett visst antal instruktioner.

Aritmetikenheten

Också aritmetikenheten (aritmetisk-logisk enhet, aritmetisk enhet) innehåller ett antal register. Enheten utför aritmetiska och logiska operationer.

Exempel på addition av två tal

1. Läs in ett värde från primärminnet och lagra det i ett register. 2. Läs in det andra värdet och lagra det i ett annat register. 3. Addera registrens innehåll och lagra resultatet i det första registret. 4. Lagra resultatet tillbaks i primärminnet

Exempel på symboliskt maskinspråk

```
0: IN R1, =KBD
1: IN R2, =KBD
2: ADD R2, R1
3: STORE R2, 7
4: MUL R2, 7
5: OUT R2, =CRT
6: SVC SP, =HALT
7:
```

Vad tror du maskinprogrammet gör?

3.6 Dataöverföring inom centralenheten.

Dataöverföring sker via ett **bussystem**, som oftast består av en **adressbuss** och en **databuss**. Dessutom behövs det en **kontrollbuss** som berättar om det är fråga om läsning eller skrivning av det data som förs över i databussen.

Via adressbussen överförs uppgifter om vilka adresser som berörs i primärminnet.

Via databussen transporteras data inom centralenheten och också till yttre enheter. Transporten sker **parallellt**. En 16 bitars databuss överför 16 bitar samtidigt. Bussens bredd är ofta densamma som datorns ordlängd.

Obs. Det finns även **övriga styrenheter** som styr kringutrustning (skrivare, mm.)

3.7 Datorarkitekturer

En dators logiska organisation kallas för dess **arkitektur**.

Arkitekturen vi just beskrivit är ett exempel på en **von Neumann-maskin**. Den kännetecknas av att

- data och program lagras i samma minne
- funktionssättet är huvudsakligen sekventiellt
- en instruktion exekveras åt gången
- en operand (ett ord) bearbetas åt gången.

För att öka prestandan har man varierat denna idé.

T.ex. data och instruktioner lagras i olika minnesbanker så att de kan hämtas samtidigt. Eller instruktioner får överlappa varandra. Då en instruktion bearbetas kan man redan läsa in den följande. Detta kallas ledning (pipelining).

Under datorernas utveckling hände det att varje gång en ny dator föddes fick man skriva om programmen. Detta så småningom ledde till **simulatorer** (som härmar effekten) eller **emulatorer** och till datorfamiljer: inom en familj sägs maskinerna vara **kompatibla** - man kan köra samma program på dem.

Maskinspråket blev mera komplicerat/abstrakt vilket gav upphov till **CISC**-datorer (Complex Instruction Set Computing). När maskinspråket blev för komplicerat uppkom en motpol i **RISC**-datorerna med många register och enkla instruktioner (Reduced Instruction Set Computer).

Det finns även andra alternativa datorarkitekturer, t.ex. datorer som är särsklit lämpade för applikationer i artificiell intelligens.

3.8 Klassificering av datorer

Mikrodatorer

- en processor, enkel struktur, 0.5-2 Mips
- ordlängd 8, 16 eller 32
- primärminne några MB
- minnesåtkomstid 100ns
- operativsystem: MS-DOS, PC-DOS, OS/2, (Linux)
- avsedda för en användare
- kan förekomma i mikrodatornätverk
- billiga, prisklass tusen till tiotusentals mk

Arbetsstationer

- en processor (RISC el. dyl.)
- ordlängd t.ex. 32
- primärminne flera MB, kanske 32 MB el. 64 MB
- operativsystem ofta UNIX

- avsedda för en användare
- kopplas till nät med större datorer
- snabba, grafik,
- prisklass: tiotusentals mk
- exempel: Sun

Minidatorer

- en processor 1-10 Mips
- ordlängd t.ex. 32, 64
- primärminne t.ex. 128 MB
- minnesåtkomsttid 50 ns
- mycket kringutrustning
- flera personer använder samtidigt
- behöver inte speciell operationspersonal
- finns mest på SM-företag (små och medelstora företag)

Stordatorer

- 2-4 processorer, ¿ 5 Mips
- ordlängd t.ex. 32, 64
- primärminne ¿ 128 MB
- minnesåtkomsttid 15 ns
- stort sekundärminne
- mycket kringutrustning
- datakommunikation
- många (t.ex 100) användare
- finns i speciella maskinrum med egen personal
- prisklass: miljoner mk

Superdatorer

- flera processorer (t.ex. 8), 200-4000 Mflops (flyttalsoperationer per sekund)
- ordlängd 64 eller 128
- minnesåtkomsttid <10ns
- specialanvändare - t.ex. väderleksprognoser
- mycket personal, specialomgivning
- prisklass: tiotals miljoner mk

Chapter 4

Datorns minnen och kringutrustning

Med kringutrustning eller perifer utrustning menas utrustning som ingår i datorsystem, avsedd för extern kommunikation eller andra uppgifter, såsom sekundärminnen och in-utenheter.

4.1 Sekundärminnen

Datorns primärminne räcker inte.

- det är av begränsad storlek
- det har högt pris
- data försvinner då strömmen bryts
- transport omöjlig utan dataförbindelser

Därför behöver vi **sekundärminne**, också benämnt **yttre minne** eller **massminne**.

Vi klassificerar minne också enligt

- dess fysiska uppbyggnad
- åtkomstprincip: hur lätt är det att läsa/skriva data från/till minnet
- datas beständighet över tid

- datas modifierbarhet

Kom ihåg att minnet storlek mäts i byte: t.ex. i kilobyte (kB), megabyte (MB), eller gigabyte (GB).

åtkomlighet

Hur många steg krävs det för att komma åt data? Detta påverkar **åtkomsttiden**.

I fall åtkomsten inte är direkt, består åtkomstiden av en **väntetid** eller **latenstid** och en **överföringstid**. Latenstiden är den tid som går åt från det att data begärs till det att överföringen kan börja.

Primärminnet är direkt åtkomligt i ett steg, sekundärminnet bara genom primärminnet.

åtkomsttider för primärminnen rör sig kring 100ns - 1 6micro;s, för sekundärminnen kanske 10 ms (t.ex. skivminnen), eller sekunder och minuter (magnetband).

åtkomstprincip

Man talar om **direktminnen**, **cykliska minnen** och **sekventiella minnen**.

I ett direkt minne, t.ex. primärminne, nås varje del av minnet på samma tid.

I ett cykliskt minne, t.ex. i ett skivminne, kan man läsa samma del av minnet vid en periodiskt återkommande tid.

I ett sekventiellt minne, t.ex. i et magnetband, kan man bara läsa data i en viss följd.

Beständighet

Man skiljer mellan **flyktiga** (obeständiga) och **beständiga** (icke-flyktiga) minnen. Data försvinner från flyktiga minne när man stänger av strömmen, medan de finns kvar på beständiga minne.

Sekundärminnen är beständiga, primärminnen (oftast) inte.

Modifierbarhet

Man skiljer mellan **permanenta** och **raderbara** minnen.

Ett permanent minne kan inte modifieras utan speciell utrustning, ett raderbart minne kan ändras och man kan lagra nya data många gånger.

WORM-minnen (Write-Once-Read-Many) är ett "präglat minne", t.ex. CD-skivor.

Typiskt permanent minne är ROM-minnet (Read-Only-Memory): datorns fasta programvara.

4.2 Olika minnestyper

Halvledarminnen

Halvledarminnen innehåller **integrerade kretsar,** lagrade på **datachips** (kretsar) eller små kiselbrickor. Dessa chips (ung. $6*6$ mm^2) kan rymma upp till och över en miljon transistorer, dvs. en miljon minnesceller. Tätheten kan överstiga 10000 bitar/mm^2.

Chipsen packas i kapslar för att skydda dem mot damm och smuts och kopplas samman på kretskort (ung. $30 * 30$ cm)

Styrenherna för kringutrustning utgörs i mikrodatorer ofta av kretskort som man kopplar in i datorn.

Magnetiska skiktminnen

Data registreras genom att ett ferritskikt magnetiseras på lagringsmediet.

Avläsning sker genom att ett **läshuvud** läser av magnetiseringen, skrivning genom att ett **skrivhuvud** magnetiserar mediet.

De vanligaste magnetiska skiktminnena är skivminne och magnetband.

Skivminnen

Skivminnet är det mest använda. Det

Figure 4.1: Skivindelning i spår och sektorer.

- är billigt
- har hög lagringskapacitet
- har kort åtkomsttid

Ett skivminne består av ett antal plana **magnetskivor** som roterar. Flera skivor bildar en **skivpacke**.

En **skivsida** är uppdelad i **spår** som är uppdelade i **sektorer**. I en skivpacke bildar motsvarande spår på olika skivsidor en cylinder. Motsvarande avsnitt på samma skivsida ingår i samma sektor (se bild).

Data organiseras i block som omfattar ett jämnt antal sektorer och är den minsta adresserbara enheten på en skiva. En fil (t.ex. ett dokument) består av ett antal

Skivminnestyp	Lagringskapacitet
Diskett	360 kB - 1.44 MB
Kassettskiva	40-80 MB
Winchesterminne	40 MB - 2 GB
Skivpacke	500 MB 2.5 GB

Table 4.1: Olika skivminnen

block som inte behöver ligga konsekutivt på skivan (skivorna).

Data läses och skrives med läs- respektive skrivhuvud. För en skivpacke förekommer det ofta lika många läs- och skrivhuvuden som det finns lagringsytor.

Disketten eller **flexskivan** används i mkrodatorer och finns i olika storlekar, 3 1/2" ("korppu") och 5 1/4" ("lerppu", äldre nu ovanligare). Tecknet " står för tum. Det finns också andra storlekar (t.ex. 8").

En **diskettenhet** består av skivaggregat som driver skivan runt samt magnethuvuden för läsning och skrivning.

Ett **Winchesterminne** (kallas också hårddisk eller **fast skiva**) används som skivminne i mindre datorer, t.ex. mikrodatorer. Kapaciteten ligger mellan 20 MB och 2 GB och åtkomsttiden mellan 10 och 30 ms.

En kasettskiva ligger inkaplsad i en plastbox. Den har diametern 5 1/4" och kapaciteten några MB.

Ett "vanligt" **skivminne** har en diameter på 36 cm. Förekommer både i skivpackar och skivkassetter (en skiva i fodral). I en skivpacke finns det 6 eller 12 skivor och de yttersta skivsidorna används inte för lagring. Sammanlagd kapacitet ca 150 - 800 MB.

Skivorna eller packarna läses och skrives i **skivstationer**.

Det finns också inkapslade skivminnen som sitter fast inmonterade med en kapacitet på 2500 MB (och diametern ca 50 cm).

åtkomstiden för skivminnen beror på följande faktorer. Skivaggregatet måste

- positionera läs/skrivhuvudet till rätt spår
- vänta tills rätt del av spåret passerar förbi
- avläsa data

Rotationshastighet för en skiva är ca 2-4000 varv/min. Positioneringstid ca 10-100 ms. Medelåtkomstid ca 20-40 ms.

Figure 4.2: Lagring på magnetband.

Obs. Ett läs/skrivhuvud måste alltid läsa/skriva i samma takt som skivan roterar.

Magnetband

Magnetband brukar också benämnas **arkivminne**. Det läses och skrives i en **bandenhet**.

Klassiska magnetband är 400, 600, 1200, 2400 eller 3600 fot långa. De består liksom magnetskivorna av ett ferritskikt som magnetiseras.

Ett tecken ligger vertikalt över bandet (över bredden) med 7 eller 9 bitar. (6+1 eller 8+1 paritetsbit).

Figure 4.3: Magnetband med och utan block.

Ett antal tecken ligger samlade i ett **block**. Före blocket kommer **blockbörjan**, efter blocket blockslut. **Blocklängden** är antalet tecken, ord eller poster i blocket.

Hastigheten ligger kring 2-5 m/s vilket ger överföringshastigheter mellan 100 kB och 200 MB per sekund. Kapaciteten kan vara mycket stor, över 100 miljoner tecken på ett band.

Alternativa tekniker: Kassettband: QIC (Quarter Inch Cassettte), Video8-kassett och DAT-band. Lagringskapacitet varierar mellan 60 och 300 MB, överföringshastighet 100 kB/s - 5 MB/s.

Band har många fördelar

- band är billiga

- band lagras lätt på hylla
- band är stryktåliga

Nackdelar är att åtkomsttiden blir lång eftersom data måste läsas sekventiellt.

Optiska minnen

Använder sig bl.a. av laserteknik.

Optodisketten (som en CD -skiva). Kapacitet 500 MB eller 1 GB.

Optoskivan (litet större än en LP). Kapacitet 4 GB.

Vissa typer kan raderas och användas på nytt.

Permanenta optiska minnen är t.ex. CD-ROM-skivor. De innehåller text, bilder, ljud, rörliga bilder. mm. t.ex. uppslagsverk. De påminner om CD-skivor (musik-skivor). Ofta kan man också spela CD-skivor i datorns CD-ROM-station.

övriga minnestyper

äldre typer av minnen: **fördröjningsledningsminnet** bestod av tankar fyllda med kvicksilver och byggde på ljudvågor som färdades i mediet!

I **trumminnnet** eller magnettrumman ligger det magnetiserbara skiktet på en cylinders yta.

Kärnminnet består av ferritkärnor (ringar på 0.5-1.5 mm) på 3-4 trådar.

Ett **bubbelminne** lagrar data i små magnetiska cylinderdomäner.

4.3 In-utenheter

In- och utenheter behövs för att mata in data och program respektive skriva ut resultat ur datorsystem.

Tangentbordet

Att skriva på ett tangentbord är ett vanligt sätt att mata in data Tangentbordet påminner om ett skrivmaskinstangentbord och man använder ofta t.ex. QWERTY-systemet för snabbare skrivning. Men det finns många extra detaljer jämfört med ett skrivmaskinstangentbord. Det finns bl.a.

- ett numeriskt tangentbord vid sidan om
- piltangenter
- funktionstangenter
- kommandotangenter

Det finns flera olika standarder och även olika språk skapar problem (accenter mm.).

Bildskärmen

Bildskärmar förekommer i många olika former. Den dominerande idag är **katodstrålerörsskärmen** (CRT display) och påminner om en TV men den har bättre kontrast och skärpa, bättre **resolution**.

Bilden genereras oftare (30-60 bilder/s) jämfört med TV (24 bilder/s). (Tänk på hur det brukar se ut när man visar en datorbildskärm på TV. Det flimrar i datarutan för att bilderna inte är synkroniserade.)

Skärmar förekommer också i olika storlekar och kvalitéer (svart-vit, färg, osv.).

Varje bildpunkt (pixel) kräver ett visst antal bitar för att representera färg, kontrast och skärpa. Från 1 bit/pixel har man utveklat bättre skärmar med 16 bitar/pixel (65536 olika möjligheter). De bästa skärmarna har 16.7 miljoner färgnyanser, dvs. 24 bitar/punkt.

Skärmar kan ge ut elektomagnetisk strålning. Nya skärmar strålar dock mindre än gamla. Kring skärmen bildas också ett elektostatisk fält (se vidare avsnittet om ergonomi).

Katodstrålerörsskärmarna är stora och tunga. En **plasmaskärm** bygger på gasurladdningsfenomen. **Flytkristallskärmen** innehåller flytande kristallin kemikalie som skapar bildkontraster. Dessa är normala i bärbara datorer.

Dataterminalen

En **dataterminal** består av tangentbord och bildskärm och utgör en ren in-utenhet utan eget minne eller beräkningsförmåga. Det finns teckenbaserade terminaler (utan möjlighet till grafik) och grafiska terminaler såsom X-terminaler.

Positionerare och pekdon

För att snabbare kunna flytta markören på bildskärmen, använder man s.k **positionerare** eller **pekdon**. Positioneraren skiljer sig från pekdonet genom att den måste kunna ange koordinatinformation. (Markören visar var bildskärmen är beredd att ta emot kommandon, t.ex. i textform, av användaren.)

Det vanligaste pekdonet är en **mus** (el. råtta). En **mekanisk mus** har en liten kula i magen. Användaren rullar musen på ett underlag och markören flyttas därefter. En **optiskt mus** har i stället en fotocell som avläser på ett underlag hur markören skall röra sig. En mus har vanligtvis tre knappar på ovansidan.

En alternativ mus är en **styrkula** på tangentbordet som roteras med tummen. Andra alternativ på tangentbordet är en liten styrkäpp (styrspak) eller en styrplatta. Styrplattan fungerar genom att användaren för pekfingret fram och tillbaka på den.

övriga styrorgan är **styrspaken** (t.ex. för spel), **ljuspennan**, **ultraljudspennan** samt **tablett** (digitaliseringsbord).

Skrivare

En **skrivare** matar ut data på papper (eller OH-film eller dylikt).

Skrivare kategoriseras enligt

- vad som skrivs i en operation
- hur det skrivs
- skrivarens styrning och tekniken bakom
- hur mycket som skrivs (på en gång)

En **teckenskrivare** skriver ett tecken i gången, en **radskrivare** en rad åt gången, en **sidskrivare** en sida åt gången.

En **anslagsskrivare** slår något (en tangent) mot pappret, en **anslagsfri skrivare** gör det inte.

Moderna skrivare läser sidbeskrivningspråk som berättar hur en sida byggs upp. Ett exempel på ett sidbeskrivningsspråk är Postscript.

Matrisskrivare

I en **matriskrivare** slungas nålar eller stift arrangerade i en kolumn mot ett färgband. Matrisen som bildar en bokstav omfattar vanligtvis 7*9 punkter, bättre kvalitet nås med högre upplösning, t.ex. 18*25 punkter.

Man kan också mata ut grafiska data, men kvaliteten blir inte så bra.

Bläckstråleskrivare

En **bläckstråleskrivare** pressar ut små bläckdroppar under högt tryck. Den är alltså anslagsfri och vanligtvis en radskrivare. Kvalitet är vanligen 360 dpi (punkter per tum). Skrivaren är lätt men långsam.

Laserskrivare

En **laserskrivare** utnyttjar samma teknik som kopiatorer. Kvalitet 300 - 1200 dpi. Laserskrivare kan styras med verklig precision. Postscript är ett allmänt använt sidbeskrivningsspråk på laserskrivare.

Värmeöverföringsskrivare

En **värmeöverföringsskrivare** bygger på att färgad vax smälts mot ett paper.

Elektrostatiska skrivare

En **elektrostatisk skrivare** fungerar enligt samma princip som en radskrivare. Utskriftsstift laddar ett papper statiskt så det drar åt sig färg. Skrivaren har hög kvalitet.

Typhjuslskrivare

En **typhjulsskrivare** är en som en elektrisk skrivmaskin.

Radskrivare

En **radskrivare** skriver en rad i gången. Hög hastighet, 1000 - 3000 rader /minut. En vanlig roman skrivs ut på ett par minuter.

Kurvskrivare

En **kurvskrivare** matar ut grafiska data. Den använder (färg-)pennor och skriver i stort format, t.ex. A0.

Optiska läsare

En bild(in)läsare (scanner) läser/kopierar in dokument (text eller bild) i rasterformat. Extra program tar hand om den inlästa bilden. Man kan t.ex. överföra inläst text inläst som bild) till text (som kan modifieras i datorn), eller ändra bildens storlek och resolution.

Andra typer av in- och utenheter

Taligenkänning: datorn försöker förstå mänskligt tal.

Talsyntes: datorn försöker tala till oss.

Dataglasögon: ögonen blir pekdon. Sensorer följer ögonens rörelser.

Datahandsken: Sensorer känner av handens rörelser.

Datahandskar och -glasögon används i **virtuella världar** (virtual reality).

Robotteknik.

Klassiska in-utenheter

Ett **hålkort** är en rektangulär skiva med 12 rader och 80 kolumner. Varje kolumn innehåller ett tecken.

En **hålremsa** är en lång pappersremsa med 5 till 8 rader plus styrhål.

övriga

Magnetbandsinkodare registrerar data från tangentbord direkt på magnetband.

Magnetisk teckenigenkänning (MICR) eller **optisk teckenigenkänning** (OCR) läser förtryckta blanketter.

Optisk märkläsning (OMR) läser markeringar på datamedium. Som exempel har vi **streckkodsläsaren** som läser EAN-koder (European Article Numbering) som finns på de flesta varor vi köper. En streckkodsläsare kan t.ex. vara en läspenna.

In- och utenheter i inbäddade system

Ett **inbäddat system** är ett datorsystem som utgör en komponent i ett större maskinellt system. Ett sådant system kräver ofta olika mätare som ger indata åt det datorbaserade systemet. Då behöver vi ofta omvandla analogt data till digitalt och vice versa.

Chapter 5

Algoritmer

En **algoritm** är en uppsättning funktionsregler för lösning av ett problem eller utförande av en uppgift i ett ändligt antal steg.

Funktionsreglerna bör vara detaljerade och entydiga och de bör ange i vilken ordningsföljd åtgärderna skall utföras.

Exempel: Att hälsa

1. Om du bär hatt, lyft den litet.
2. Buga lätt.
3. Om du lyfte på hatten, sätt tillbaka den på huvudet.

Men t.ex. regeln ”Hedra din fader och din moder” är ingen algoritm.

Varför behöver vi algoritmer?

1. Datorn fungerar algoritmiskt.
2. De flesta programspråk är algoritmiska.
3. Användningen av databehandlingssystem beskrivs ofta algoritmiskt.

Svårighet: Människans tänkesätt är inte algoritmiskt.

Tröst: Bitar är ännu mera avlägsna från människans tankesätt.

Förutsägelse: Det blir bekvämare att använda datorer.

Exempel: Att klättra upp på taket.

Sekventiella åtgärder	1. Duscha.
	2. Klä på dig
Villkorlig åtgärd	3. **Om** kvällen blev sen, tag en jaffa ur kylskåpet **annars** koka kaffe.
Upprepad åtgärd	4. **Så länge** du är hungrig: ät en smörgås och drick en klunk.
Sekventiell åtgärd	5. Gå till busshållplatsen.

Table 5.1: Exempel: Morgonprogram.

1. Kliv upp på den första stegpinnen.
2. Så länge som det finns stegpinnar kvar ovanför dina fötter, kliv upp på följande pinne.
3. Kliv från den översta stegpinnen till taket.

Detaljnivån i en algoritm beror på hur mycket den som skall utföra algoritmen förstår.

- För en dator ges algoritmen som mycket detaljerade datorprogram.
- En människa förstår med mindre.

Algoritmer och algoritmiskt tänkesätt behövs vid utvecklingen av program. Utvecklingsarbetet kan indelas i sju skeden:

1. Definition av uppgiften.
2. Definition av programspecifikation.
3. Planering av programmet.
4. Programmering.
5. Testning av programmet.
6. Ibruktagning och underhåll.
7. Dokumentering.

I algoritmer används tre olika grundstrukturer. Med hjälp av dessa kan alla algoritmer presenteras.

Presentationssätt

- Numrerade steg
- Traditionella funktionsscheman
- Strukturerade funktionsscheman

Numrerade steg

Se ovan.

Traditionella funktionsscheman

Se bild.

Strukturerade funktionsscheman

Se bild.

5.1 Algoritmer som modell för datorprogram

Algoritmerna i våra exempel hittills har varit avsedda för människor. Därför kunde abstraktionsnivån vara så hög att en dator inte har någon möjlighet att fungera enligt dem.

Ett **datorprogram** är en algoritm som är så exakt uttryckt att t.o.m. en dator kan fungera enligt den.

Lösning av problem med datorprogram:

Programmering (Uppgift → Algoritm → Program) → översättning (Maskininstruktioner) → Exekvering (Resultat)

Nödvändiga begrepp i datoralgoritmer är

- variabel
- läsoperation
- skrivoperation

En **variabel** är ett förvaringsställe för ett värde. En variabel har ett **namn** och en **typ**. Typen anger hurudana värden variabeln kan få.

Exempel:

En variabel kan ges ett värde genom en **tilldelningsoperation**.

Figure 5.1: Presentationssätt för algoritmer.

Figure 5.2: Hälsningsalgoritm med traditionellt funktionsschema.

Variabelns typ	Möjliga värden
Heltal	..., -1, 0, 1, ...
Realtal	10.0, 0.05, 2.326
Teckenföljd	abrakadabra, 4711, B450
Logiskt värde	sant, falskt

Figure 5.3: Hälsningsalgoritm med struktureat flödesschema.

Figure 5.4: Klättringsalgoritm med traditionellt funktionsschema.

Figure 5.5: Klättringsalgoritm med struktureat flödesschema.

DUSSIN := 12

Ovanstående operation betyder att variabeln med namnet ”DUSSIN” får värdet 12. Men lika gärna kunde vi skriva ”DUSSIN := 20”.

En algoritm kommunicerar med sin omgivning genom **inmatnings-** och **utmatningsoperationer**.

Exempel:

INDATA *longrightarrow* ALGORITM *longrightarrow* UTDATA

In- och utdata konkretiseras som en **kö** av värden.

Exempel:

Innan algoritmen utförs:

5, 13, 17, 3, 1 *longrightarrow* ALGORITM

Efter att algoritmen utförts:

ALGORITM *longrightarrow* 39, 5, 8

En inmatningsoperation (eller läsoperation) har formen

LäS (VARIAB) eller **LäS (VARIAB1, VARIAB2)** osv.

där VARIAB, VARIAB1, VARIAB2 är namn på variabler.

LäS() åstadkommer följande:

- Följande variabel i variabelförteckningen tilldelas följande värde i inkön.
- Ett en gång inläst värde kan inte läsas på nytt.
- Ett inläst värde kan lagras som värdet av någon variabel.

En utmatningsopertaion (eller skrivoperation) har formen

SKRIV (UTTR) eller **SKRIV (UTTR1, UTTR2)** osv.

där UTTR, UTTR1, UTTR2 är uttryck. Som uttryck duger t.ex.

- Ett variabelnamn

Figure 5.6: Vad beräknar algoritmen?

- En konstant (numerisk: 3.1416, 1997 eller teckensträngskonstant ”4. Algoritmer”, ”oktober”, ”abc”)
- Egentliga uttryck (2 * PI * R + konstant)

SKRIV() åstadkommer följande:

- Värdet av följande uttryck i uttrycksförteckningen skrivs ut på följande ställe i utkön.
- När värdet är utskrivet kan det inte mera tas tillbaka.

Exempel:

Läs namn	Antal kvar	Läs namn	Antal kvar
1	50000	1	1600
1	25000	1	800
1	12500	1	400
1	6300	1	200
1	3200	1	100
	100	0	

En algoritm som beräknar och skriver ut en rektangels yta och omkrets, då längderna av rektangelns sidor ges som indata.

Uppgift: Inkön innehåller tal (minst ett). Skriv en algoritm som skriver ut det största av talen.

Extra övning: Ge ovanstående algoritm med numrerade steg och som ett traditionellt funktionsschema.

En given uppgift kan vanligen lösas med flera olika algoritmer.

Algoritmerna kan vara bättre eller sämre, dvs. de kan jämföras med varandra i fråga om

- Användning av tid.
- Användning av utrymme.

Jämföringen kan göras

- I genomsnitt (genomsnittsligt indata).
- I värsta fall (värsta indata).

Exempel: Sök ett nummer i en telefonkatalog med 1000 sidor och 100 namn/sida.

Algoritm 1: Sökning i ordningsföljd från början. I värsta fall är det sökta namnet sist i katalogen, dvs. man måste läsa 1000*100 = 100 000 namn.

Algoritm 2: öppna katalogen i mitten och välj den halva som innehåller det sökta namnet. Upprepa halveringen tills en sida är kvar. Sök namnet i ordningsföljd på denna sida.

I värsta fall:

Allt som allt läses 110 namn vilket är mycket mindre än 100 000.

Parallell funktion och obunden ordningsföljd

En dators funktion är sekventiellt algoritmisk eftersom processorn utför operationerna sekventiellt.

Exempel: I följande tilldelning beräknas först uttrycket till höger

X := A * B + C * D - E / 2 - 4

på följande sätt:

1. A*B
2. C*D
3. E/2
4. A*B + C*D
5. A*B ? C*D - E/2
6. A*B + C*D - E/2 - 4
7. X := A*B + C*D - E/2 - 4

En del operationer kan utföras parallellt om det finns flera processorer i datorn.

Parallellt

1. A*****B, C*****D, E/2
2. A*B **+** C*D, E/2 **-** 4
3. A*B + C*D **-** E/2 - 4
4. X **:=** A*B + C*D - E/2 - 4

De uppgifter och behandlingsregler som behövs för att lösa ett problem i en sekventiell algoritm kan lagras i en dators minnesenheter i obunden ordningsföljd som fakta och behandlingsregler för dessa.

En in minnet lagrad inferensmekanism (ett program) kombinerar uppgifter ur kunskapen enligt en användares förfrågningar.

Användare <*longrightarrow* inferensmekanism <*longrightarrow* kunskap

5.2 Algoritmers begränsningar

Ett algoritmiskt problem specifierar en mängd tillåtna indata och en utdatamängd som en funktion av dessa indata. Problemet är löst när man funnit en algoritm som givet godtyckliga tillåtna indata producerar utdata tillhörande enligt den specifierade funktionen.

Vi frågar oss:

- Finns det en användbar algoritm för detta problem?
- Finns det överhuvudtaget en algoritm för detta problem?

Svaret är ibland "Ja", ibland, "Nej", ibland "Vet inte".

Det finns alltså problem där man vet att det inte finns någon användbar algoritm, och andra problem där man tror, men inte vet om det finns någon algoritm.

På motsvarande sätt. Det finns problem som man vet att inte har någon algoritmiskt lösning och problem där man fortfarande söker efter en möjlig algoritm.

Man kan dela upp de algoritmiska problemen enligt komplexitet: **hanterliga**, **ohanterliga** och **oavgörbara**.

Hanterliga problem

Hanterliga problem (tractable problems) har kända algoritmer med polynomiell tidskomplexitet, dvs. antalet operationer som funktion av problemets storlek **n** kan uttryckas som ett polynom i **n**.

Ohanterliga problem

Ohanterliga problem (intractable problems) har algoritmer som kräver så stora resurser att de är oanvändbara.

Exempel: Tornen i Hanoi

(T.ex. de s.k. **Np-kompletta** problemen misstänks vara ohanterliga.)

Oavgörbara problem

Ett algoritmiskt problem är **oavgörbart** (ratkeamaton, undecidable) om man kan bevisa att det inte kan existera någon algoritm för problemet.

Exempel: Halting-problemet: Finns det en algoritm som kan ta en godtycklig algoritm och godtyckliga indata till denna och avgöra om denna algoritm kommer att terminera eller ej för dessa indata? (Terminera = avsluta utförandet.)

Påstående: Nej

Bevis:

Antites: Vi antar att det finns en algoritm M_1 som som läser två godtyckliga indata **M** (en algoritm) och M' (dess data). Vi betecknar $\mathbf{M}_1(M, M')$ = resultatet då algoritm M_1 utförs med indata M och M'.

Algoritmen M_1skriver ut värdet

- sant, om algoritmen M terminerar för indata M'
- falskt, om algoritmen M inte terminerar för indatat M'

Då kan vi göra upp en algoritm M_2 som innehåller instruktionen:

Om $M_1(M, M')$ **så gå i oändlig slinga (= terminera inte), annars skriv ut värdet falskt.**

Då får vi en motsägelse eftersom:

$M_2(M_2)$ terminerar $\iff$ $M_1(M_2, M_2)$ = falskt $Longleftrightarrow$ $M_2(M_2)$ terminerar inte.

Chapter 6

Programspråk

Programspråk (el. programmeringsspråk) behövs för att beskriva algoritmer exakt för en dator. Programspråkens viktigaste egenskap är att de är **exekverbara**, dvs. datorn kan utföra de instruktioner som programspråken uttrycker.

Det är svårt att ge en översikt över existerande programspråk efter som det finns så många. Vi skall i alla fall nämna de viktigaste inom olika språkkategorier.

6.1 Programspråksutvecklingen

Programspråken kan kort indelas i

- maskinspråk (1:a generationen)
- assemblerspråk (2:a generationen)
- högnivåspråk (3:e generationen)
- applikationsspråk (4:e generationen)
- språk för artificiell intelligens (5:e generationen)

Man kan generellt säga att språken utvecklats från enklare (maskinspråk) till mera komplicerade/abstrakta/användarvänliga. Dock används fortfarande språk ur alla ovanstående kategorier. Märk även att högnivåspråken rymmer många olika språk (och är kanske de vanligaste).

En finare uppdelning av språk delar in dem enligt **programmeringsparadigmer**, dvs. en övergripande föreställning om hur man bygger programsystem, vad de avbildar och deras struktur.

Familj/paradigm	Exempel
Imperativa språk	Fortran, Cobol, Pascal, C, Modula-3 och många andra
Objektorienterade språk	Simula, Smalltalk, C++, Object Pascal, Eiffel
Funktionella språk	Lisp, Scheme, Standard ML
Parallella språk	RTL/2, Concurrent Pascal, Chill, Ada, Occam
Logikspråk	Prolog

Språken har dock gemensamma drag även om de följer olika paradigmer, t.ex. de flesta språk har

- in- och utmatningsoperationer
  - enhet
  - datats form
- de enkla räknesätten +, -, *, / (addition, subtraktion, multiplikation, division)
- möjlighet att avvika från det normala sekventiella utförandet av satserna
  - förgreningssatser
  - villkorliga och upprepade strukturer
  - underprogram
- tilldelningssatser
  - data från/till primärminnet
- möjlighet att förse programmet med kommentarer i klartext

Utvecklingen har strävat till att införa bregepp som

- är maskinoberoende,
- stöder människans sätt att tänka,
- ansluter sig till en viss programmeringsparadigm och
- uttrycker funktioner i nya tillämpningar på ett acceptabelt sätt.

Begreppen bör vara generella, kraftfulla, tillämpningsnära och paradigmanknutna och dessutom låta sig implementeras effektivt.

6.2 Abstraktion

Man strävar genom programspråk alltså till att **abstrahera** från datorns egenskaper och närma sig människans tänkesätt.

Detta har man lyckats med genom ett antal s.k. abstraktionsmekanismer

- **procedurell** och **funktionell abstraktion** - man löser delproblem skilt för sig i **underprogram** som kan bestå av **funktioner**, **procedurer** eller **funktionsprocedurer**. Från andra platser i programmet kan man sedan anropa underprogrammet när deluppgiften skall utföras. I procedurell abstraktion kan även åstadkommas med s.k. **makrodeklarationer**, programavsnitt som genom **makroinstruktioner** i programmet bara kopieras in på rätt ställe när programmet översätts.
- **uttrycksabstraktion** - tillåter att matematiska och logiska uttryck kan skrivas på sätt som ligger nära de sätt människan använder.
- **flödesabstraktion** - tillåter att programmet "flödar", dvs. instruktionerna utförs i en naturlig ordning enligt de sammansatta satser, villkorliga och upprepande statser som de bstår av. Motsatsen är t.ex. Fortran och Basic som innehåller GOTO-satser.
- **dataabstraktion** - användaren kan definiera sina egna data och ge dem namn efter tillämpningen och problemet.
- **datatyps-** och **objektsabstraktion** - användaren kan definiera sina egna datatyper och och ge dem namn efter tillämpningen och problemet. En abstrakt datatyp (ADT) innehåller en datastruktur och de metoder som får tillämpas på de data som ingår. I objektsabstraktionen definieras en klass (med datastrukturer och metoder) samt objekt som tilhör denna klass..

6.3 Språkdefinition

I ett programspråk definieras

- **syntax** (exakta grammatikregler), och
- **semantik** (betydelsen av uttrycken)

Syntaxen säger alltså hur korrekta program får se ut, semantiken vad som krävs för att korrekta program ska vara meningsfulla och vad som ska hända när de översätts och utförs.

I bland blir det fel i ett program. Ett **kompileringsfel** upptäcks då programmet översätts till maskinspråk och kan bero på brott mot syntaxen (vi har gjort skrivfel eller placerat instruktionsdelar i fel ordning el. dyl.) eller på den statiska semantiken (vi har glömt att ange typen för en variabel).

Under utförandet av programmet kan det också förekomma fel. Brott mot den dymaniska semantiken inbegriper t.ex. att vi försöker dividera med talet noll. Dessa fel kallas också **exekveringsfel**.

Ett **logiskt fel** gör att programmet producerar felaktiga resultat. Eller att det egentligen löser ett annat problem än vad det var menat för.

6.4 Programspråkskategorier

Maskinspråk

Maskinspråk är det enda språk som datorn förstår. Alla andra programspråk måste översättas till maskinspråk.

Maskinspråksinstruktionerna uttrycks som bitföljder med en operationsdel och en adressdel, t.ex.

00010110 0011000

öKR 24

Maskinspråk utgjorde den enda programmeringsmöjligheten i de första datorerna.

Maskinspråk är **maskinberoende**.

Idag är det ovanligt att man programmerar direkt på maskinspråk men man kan göra det, t.ex. för att optimera (effektivera) någon del av ett program.

Assembleringsspråk

Assembleringsspråk kallas också **symboliska maskinspråk**.

Maskinspråksinstruktionerna ersätts med symboliska namn och gör därmed maskinspråksprogrammeringen lättare.

Ett program som översätter assembleringsspråk till maskinspråk kallas **assemblator** (el. assemblerare)

Ett assembleringsspråk är (ofta) maskinberoende.

Exempel (skrivet på Extrasimpel, ingår inte i kursen):

1. LäR
2. LGR 10
3. LäR
4. öKR 10
5. SKR
6. STP

Högnivåspråk

Högnivåspråk är **problemcentrerade** och **maskinoberoende** i stället för maskincentrerade. Det är lättare att programmera och programmen blir tillförlitligare. Det finns t.ex. mekanismer för

- matematiska uttryck
- villkorliga och upprepade strukturer
- underprogram
- strukturerad programmering

Maskinoberoendet gör alltså att man kan köra samma program på olika maskiner. Programmen måste dock skilt för sig **översättas** (kompileras) på respektive maskiner.

Exempel på högnivåspråk

- BASIC (Beginner's All-purpose Symbolic Instruction Code)
  - utvecklades 1965
  - ursprungligen enkelt och lätt att lära sig
  - förr populärt i mikrodatorer
  - s.k. GOTO-instruktioner kunde göra spagetti av stora program
- FORTRAN (FORmula TRANslator)

  - det äldsta högnivåspråket (1957) fortfarande i användning?
  - väl standardiserat
  - matematiskt inriktat, t.ex. numerisk precision. mm.
  - besvärliga in- och utmatningsinstruktioner

- COBOL (COmmon Business-Oriented Language)
  - utvecklades 1960
  - behandling av data i teckenform
  - besvärliga kontrollstrukturer - ¿ programmen blir långa och språket olämpligt för invecklade uppgifter
  - (var) viktigt inom handel och administration

- ALGOL (ALGOrithmic Language)
  - Pascals föregångage på 1960-talet
  - betydande påverkan på andra språk
  - undervisningspråk före Pascal

- Pascal (efter Blaise PASCAL)
  - uppfanns 1971 av schweizaren Niklaus Wirth
  - gjort med tanke på strukturerad programmering
  - utmärkt tidigare undervisningsspråk: leder till goda programmeringsvanor
  - princip: det är bättre att ett felaktigt program inte gör någonting alls än att det åstadkommer förödelse

- MODULA (MODULar LAnguage)
  - utvecklades 1978 av Niklaus Wirth som Pascals efterträdare
  - poängterar modularitet

- C
  - utvecklades vid Bell Laboratories tillsammans med operativsystemet UNIX av Dennis Ritchie 1972
  - möjlighet till maskinnära programmering
  - tillåter allt möjligt och ger därmed möjlighet till svårupptäckta fel

- ADA
  - introducerades 1980
  - liknar MODULA-2 (poänterar strukturerad programmering)

  - innehåller massor med egenskaper -¿ svårt att lära sig

- C++
  - objektorienterad C
  - populärt idag

- Java
  - utvecklat nu på 1990-talet
  - baserat på C++, objektorienterat
  - stora programbibliotek med färdigt programmerade funktioner
  - används bl.a. på internet: en s.k. **applet** (programmerad i Java) kan köras i webbläddraren

Applikationsspråk

Applikationsspråk försöker göra det lättare för programmeraren/användaren genom sin abstraktion. Applikationsspråk är specialspråk för olika tillämpningar, t.ex.

- **frågespråk**, t.ex. i databassystem Ett exempel är SQL med vilket databasers information behandlas.
- **generatorer för användargränssnitt**. Programmeraraen skapar ett grafiskt användargränssnitt genom att plocka ihop olika fönster/knappar och menyer som skall ingå i gränssnittet. Gränssnittet kan testas direkt.
- övriga **generatorer**.

Applikationsspråk kan vara grafikbaserade.

Språk för artificiell intelligens

Språk för artificiell intelligens är ofta **definierande**: man beskriver mindre **hur** ett problem skall lösas och mera **vad** problemet består av.

Exempel på språk för artificiell intelligens:

- LISP (LISt Programming)
  - utvecklat på MIT av John McCarthy 1960

  - grunddatastrukturer: lista
  - lämpligt för applikationer i artificiell intelligens

- PROLOG (PROgrammation et LOGique)
  - utvecklat av Philippe Rossel vid universitetet i Marseille i början av 1970-talet
  - logikprogramspråk
    lämpligt för expertsystem (artificiell intelligens) och hantering av naturliga språk

6.5 Kriterier för val av programspråk

Då vi väljer vilket programspråk vi skall använda för en speciell applikation bör vi beakta

- språkets lämplighet (inte COBOL för numerisk matematik)
- språkets tydlighet och svårighetsgrad, dvs. att det är lätt att skriva program på det språket
- effektivitet (snabb kompilering och körning, effektiv minnesanvändning, tillräcklig uttryckskraft så att programmen inte blir onödigt långa utan kan skrivas snabbt)
- tillgänglighet och stöd (endast sådana språk som är tillgängliga kan användas; om språkets användning/utveckling sinar är det inte lönt)
- konsistens med tidigare programvara (lätt att hålla sig till språk andra program är skrivna på)

Chapter 7

Programvara

7.1 Systemprogram vs. applikationsprogram

Det är långt mellan maskinens behandlingssätt (maskinspråk) och människans begreppsvärld (naturligt språk).

Utan **program** kan en dator inte göra något (lika litet som utan elektricitet).

Programvaran (samling program, alla program, mjukvara) utgör förbindelsen mellan människan och datorn.

Systemprogram finns till för att göra det lättare att styra datorns funktioner. De tillhandahåller basoperationer som används t.ex. när applikationsprogram utvecklas.

Viktiga systemprogram: operativsystem, kompilatorer.

Applikationsprogram (även tillämpningsprogram) är avsedda för utförandet av enskilda uppgifter som **primäranvändaren** vill utföra.

Traditionellt kan man säga att systemprogrammen används främst av ADB-yrkesmänniskor, applikationsprogrammen av användare utan omfattande ADB-skolning. Men användningen varierar.

7.2 Systemprogram

Det viktigaste systemprogrammet är **operativsystemet**.

Varje dator måste ha ett operativsystem för att den skall gå att använda. (Ibland kan man byta mellan olika operativsystem i samma fysiska maskin.)

Operativsystemet består av en samling program som kontrollerar, övervakar och stöder användningen av maskinvaran. Operativsystemet sköter bl.a. följande uppgifter:

- filhantering - systemet lagrar och raderar filer, ordnar filerna, kopierar, flyttar
- kringutrustning - systemet underlättar användningen av denna, in- och utmatning, datakommunikation
- i system med flera användare fördelar systemet datortiden mellan olika användare
- tidsdelning - om flera processer (program) körs samtidigt fördelar systemet turerna
- säkerhet - systemet sköter om användaridentifikation och lösenord
- användargränssnitt - systemet fungerar som ett användargränssnitt till datorn

Operativsystemets basprogram kallas **kärna**.

Filhanteringen sköts i regel med en trädstruktur, där filerna är samlade i hierarkiskt ordnade **kataloger**. Baskatalogen kallas rot. Rätt fil nås längs med en sökväg i trädstrukturen. Katalogen man arbetar i kallas **arbetskatalog** (aktuell katalog) och en användares rotkatalog **hemkatalog**.

Begrepp som gäller operativsystem

Multiprogramkörning innebär att användare kör flera program samtidigt. Då vi har fler program i en dator och datorn ger varje användare ett litet tidsintervall CPU-tid i tur och ordning talar vi om **tidsdelning**. Alla användare har illusionen av att vara den enda användaren (om det inte finns för många användare).

I större datorer kan man också utföra **satsvis bearbetning** som innebär att programmen sätts i kö och behandlas vid senare tidpunkt, t.ex. på natten när användningen är liten.

Multiprocessorsystem sköter samtidigt om körning av flera program (eller delar av ett program) på en dator med flera processorer, s.k. **multiprocessor**.

Realtidssystem behövs t.ex. inom processindustrin: registreringsapparatur förmedlar uppgifter om processens tillstånd och uppgifterna behandlas omedelbart och processen kan påverkas beroende på resultaten.

För multiprogramkörning används numera allmänt **fönsteromgivningar** (t.ex. Windows). Olika applikationsprogram har också användargränssnitt som baserar sig på användningen av fönster.

För fönsteromgivningar krävs en bildskärm med grafikegenskaper. Program och andra filer representeras av **ikoner** som kan öppnas eller stängas. Kommandon utförs genom att man väljer instruktionerna i **menyer**.

Program som diskuterar med användaren kallas **interaktiva**. Interaktiva program behöver inte basera sig på fönsteromgivningar utan användargränsnittet kan vara textbaserat.

övrig systemprogramvara

övriga systemprogram kommer till användning vid programutveckling:

- Först skrivs programmet med en redigerare (editor).
- Sedan **översätts** det till maskinspråk av en **översättare**.

En **översättare** är ett program som omvandlar kod från ett språk till ett annat.

översättning från assembleringsspråk till maskinspråk kallas **assemblering** och utförs av en **assemblator** (**assemblerare**).

översättning från ett högnivåspråk till maskinspråk kallas **kompilering** och utförs av en **kompilator**.

Man startar med ett **källprogram** som kompileras till ett **målprogram**.

Stora program kan delas i moduler som kompileras var för sig och **länkas** (kombineras) av en **länkare** till ett utförbart prorgram.

Program innehåller ofta fel som man försöker hitta genom att **testa** programmet. För att söka fel kan man använda en **debugger** (avlusningsprogram).

Det finns också ett med kompilering alternativt sätt att utföra program på högnivåspråk: **tolkning**. En **tolk** översätter och utför varje sats i ett källprogram innan nästa sats översätts och utförs.

7.3 Användargränssnitt

Ett **användargränssnitt** behövs i alla program.

Användargränssnittet låter användaren mata in data och visar resultat. För många användare är användargränssnittet det samma som själva programmet!

Ett användargränssnitt kan baseras på

- **textuella kommandon**, t.ex. MS-DOS och UNIX. Kommandon ges som text med olika parametrar och systemet skriver ut svaret. **Ledtexten** (ledsymbolen, promptern) visar (och uppmanar) användaren var kommandot skall skrivas in. **Markören** blinkar eller visar var texten kommer att dyka upp på skärmen.
- **menyer**. Alla funktioner syns och är valbara i en **textuell meny** eller en **rullgardinsmeny**.
- **blanketter**. Bildskärmen visar blanketter med tomma rader/fält som användaren fyller i.
- **grafik**, t.ex. Windows, Netscape, m.fl. Användaren använder **mus** men även både menyer och blanketter. I gränssnittet ingår ofta **ikoner** som representerar program, filer, alternativ, mm.

7.4 Problemet med år 2000

Vi närmar oss år 2000 (litet över 2 år kvar).

I datavärlden/affärsvärlden är man rädd för något som man ibland kallar ”the year 2000 problem” eller kort och gott ”the Y2K problem”.

Problemet kan uppstå då man i tidigare program använt endast två siffror för att koda årtal. Man tänkte sig inte att man någonsin skulle behöva använda hela årtalet. Detta har man gjort för att

- spara utrymme (två siffror tar mindre plats än fyra och förr var det brist på minnesutrymme)
- spara tid (det är effektivare att räkna med tvåsiffriga tal än med fyrsiffriga

Ifall man använt bara två siffror för årtalen i ett program, kommer detta århundrades sista år att vara kodat ”99” och efter nyår går man över till året ”00”, dvs. ett årtal som matematiskt står för ett mindre tal. Tiden tycks ha gått bakåt.

Gammalt kan verka ungt, vissa ögonblick tycks ha tagit ett sekel, framtiden har redan passerat.

Exempel:

Kalle har jobbat från den första december 1999 till den sista januari 2000. Hans månadslön är 10000 mk. Lönesystemet har kodat årtalen med bara två siffror. Således har Kalle börjat jobba 99-12-01 och slutat 00-01-31. Systemet räknar nu den tid Kalle jobbat genom att subtrahera den sista dagen från den första för att få den sammanlaga arbetstiden.

Om vi minskar året ”00” med året ”99” får vi naturligtvis -99. Det kan hända att maskinen märker att detta tal inte är möjligt (negativt, och eller för stort). Det kan också tänkas att systemet endast tittar på absolutbeloppet och kommer fram till att Kalle jobbat ca 99 år varvid hans lön kommer att bli en så där 12 miljoner mk.

Om lönesystemet kodat årtalen med fyra siffror skulle naturligtvis samma problem inte uppstå, för ”2000-01-31” - ”1999-12-01” = 2 månader och lönen blir (tråkigt nog) 20000 mk.

Exempel 2:

Wilhelmina Josefina Karlsdotter är jämngammal med det 20:e århundradet, dvs. hon är född den första januari 1900, kodat 00-01-01. På våren 2006 får hon plötsligt en inbjudan att börja förskolan nästa höst. Varför?

Följderna kan naturligtvis bli allvarligare om berörda program är livsviktiga, t.ex. används för att styra apparater och maskiner såsom flygmaskiner, medicinsk apparatur osv.

Det är emellertid svårt att uppskatta hur allvarligt problemet är.

De flesta program torde vara korrekt uppgjorda. Men faktum kvarstår att det finns många program som måste uppdateras och rättas om de skall fungera rätt efter den första januari 2000.

Datakonsulter och -programmerare jublar ... Pengarna strömmar in ...

Vink: Problemet är känt, det är relativt lätt att rätta och det finns fortfarande tid att rätta till det, men kanske bör man undvika att befinna sig i ett flygplan på nyårsnatten mot år 2000...

Chapter 8

ADB-historik

8.1 Urtid

Först kunde människan bara skilja mellan en/ett och många, senare kom begreppen två, tre, osv.

Man representerade antalet genom att helt enkelt rita av det föremål som man räknade, t.ex. **grottmålningar** i Altair (?).

Inristningar på ben för ca 30000 år sedan visar hur man räknade boskap: Ett streck för varje djur. Ben var lättare att hantera och bära med sig än en grottmålning.

Så småningom började man **räkna i grupper** på fem (romarna) och tio (Indien, Arabien).

Den första räknemaskinen var **räknebordet** för ca 3000 år sedan. På bordet representerade stenar tal.

Räknebordet utvecklades så småningom tusen år senare till en **kulram**, portablare (bärbar) än bordet.

Algoritmiskt tänkesätt i bl.a astronimiska beräkningar från Babylonien för 4000 år sedan.

Axiomatisk matematik i antikens Grekland.

På 800-talet var Kalifatets huvudstad Bagdad ett viktigt centrum för den tidens vetenskap:

- Efter Al-Khwarizimi, matematiker, har vi döpt både **algebra** och **algoritmer**

- Den antika vetenskapen levde vidare genom arabiska kopior till dagens tid.

8.2 Medeltid och upplysningstid

Astronomer hade analoga redskap.

De första digitala räknemaskinerna uppfanns på 1600-talet

Wilhelm Shickard (1592-1635) byggde en additionsmaskin.

Blaise Pascal (1623-62) byggde en addition-subtraktions-maskin (Pascaline) som fungerade med kugghjul.

Gottfried Wilhelm von Leibniz (1646-1716) byggde en multiplikations-divisionsmaskin (som också kunde addera och subtrahera).

Representation av digitala data i olika automater såsom speldosor och kyrkklockor

Joseph Marie Jacquard (1752-1834) uppfann en vävmaskin som styrdes av hålkort.

8.3 1800-talet

Charles Babbage (1792-1872) uppfann en **analytisk maskin** (analytical engine), men uppfinningen var tekniskt för svår för att man skulle kunna bygga datorn på 1800-talet. Babbages dator var mekanisk, hade mekaniskt minne och använde hålkort. (På senare tid har man byggt en modell efter Babbages ritningar.)

Ada Augusta Byron, Lady Lovelace (1816-52, dotter till Lord Byron) utvecklade Babbages ideer och skrev de första datorprogrammet (1842). Hon anses vara den första programmeraren då hon bl.a. uppfann den upprepade åtgärden (en s.k. loop) och den villkorliga åtgärden. (Obs. Programspråket Ada har döpts efter henne.)

William Seward Burroughs (1855-98) byggde en kommersiell additionsmaskin. Redan 1926 hade man konstruerat en miljon sådana maskiner.

Dr. **Herman Hollerith** (1860-1929) som arbetade på USAs folkräkningsbyrå (US Census Office) byggde en elektrisk hålkortstabulator och grundade Tabulating Machine Company. 1924 bytte bolaget namn till International Business Machines, IBM). Chef var då Thomas J. Watson Sr. (1874-1956).

- 1880 hade man hållit en folkräkning som det tog sju år att manuellt slutföra.

- 1890-års folkräkning skulle med samma metoder ta mer än tio år, men följande folkräkning skulle ske redan åt 1900.
- Tack vare Holleriths maskin gjordes räkningen på tre år.
- På hålkorten slog man in uppgifter, inte bara om antal människor utan också antal barn, hemspråk, osv.
- Hålkortstabulatorn kunde sortera hålkorten och det var lätt att räkna t.ex. hur många familjer det fanns med fler än två barn.

8.4 Början av 1900-talet: De första datorerna

1936-38 byggde tysken **Konrad Zuse** den första binära digitala elektroniska datorn Z1. Den hade ett mekaniskt minne. (Men fungerade den?)

Andra världskriget satte fart på utvecklingen.

1941-3 utvecklades i England en elektronisk räkneapparat (Colossus) av **Alan Turing** för att lösa tyskarnas chiffersystem. Den hade ett minne som baserade sig på reläer, senare på elektronrör. Alan Turing definierade redan 1936 en abstrakt generell modell för en datamaskin, den s.k. **Turingmaskinen**.

Mark I-datorn i Harvard blev färdig 1944. Den var baserad på elektriska reläer. En multiplikation tog 3 sekunder. Mark II var den första maskinen som man hittade en 'bugg' (bug, programfel) i. I själva verket var den första **buggen** en mal som fick maskinen att stanna. Därefter skyllde teknikerna på att de höll på att debugga (avlusa) datorn om den inte fungerade! (Anekdot med något tvivelaktig sanningshalt: Ordet **bug** förekommer redan på 1800-talet i Oxford English Dictionary med betydelse fel i maskin. thomas Alva Edison lär ska vara den första som använde ordet i den betydelsen 1878.)

år 1946 kom ENIAC (Electronic Numerical Integrator and Computer) som var en helt elektronisk maskin. Den var 30 m * 1 m * 3 m , vägde 30 ton och innnehöll 18000 elektronrör. Programmering innebar att man gjorde omkopplingar i maskinens kabelsystem. Maskinen hade ett primärminne med 20 register där varje register rymde ett tiosiffrigt decimaltal. En multiplkikation tog 3 ms. ENIAC användes bl.a. för beräkningar i vätebombsutvecklingen..

John von Neumann (1903-57) formulerade datorn av idag med principen om program lagrade i minnet (EDVAC-maskinen, Electronic Discrete Variable Automatic Computer). EDVAC innehöll processor, aritmetik-logikenhet och primäminne. I produktion kom den först 1951 pga av förseningar. (Egentligen var det John W. Mauchley och J. Presper Eckert som kom på idén om program lagrade i minnet, bäst vore det att prata om EDVAC-principen.)

Den första maskinen (som blev klar) som använde principen med program i minnet var EDSAC (Electronic Delay Strage Automatic Calculator) i Cambridge 1949.

Den första maskinen i USA som byggde på ovannämnda princip var BINAC (Mauchley & Eckert) som blev klar också den 1949. Tyvärr fungerade den aldrig sedan den levererats till kunderna. Den hade det första högnivåspråket **Short Code**.

IAS (v. Neumann) från år 1952 var den första generella datorn. Användes bl.a. vid utveckling av kärnvapen.

8.5 Första generationen datorer: 1951-58

De första kommersiella datorerna kom med i bilden från ca 1951.

Whirlwind, FSQ-7, IBM701, IBM702...

Karakteristika: elektronrör, hålkort, och magnetiskt primärminne. I slutet av perioden kom också magnetband, programmering med maskinspråk och assembleringsspråk in i bilden.

I Finland byggdes den första datorn ESKO på Tekniska högskolan 1958.

Den första kommersiella datorn i Finland togs i bruk i Postbanken 17.10.1958 (IBM 650).

8.6 Andra generationen datorer: 1959-64

Den andra generationen baserade sig på transistorn som man uppfunnit redan 1948.

Massproduktion inleddes 1959.

Karakteristika: transistorer, magnetiska skivor, högnivåspråk (FORTRAN, COBOL).

8.7 Tredje generationen datorer 1965 -71

IBM:s datorfamilj IBM 360 (kompatibla).

Karakteristika: integrerade kretsar (1969: 100 transistorer/kiselbricka), dataskärmsterminaler, utbytbara skivor, nya högnivåspråk (BASIC), operativssystem, tidsdelning

De första datalogiprofessurerna (och institutionerna) i Finland

- 1965 Tammerfors universitet
- 1967 Helsingfors universitet
- 1967 Jyväskylä universitet
- 1968 Tekniska högskolan

8.8 Fjärde generationen datorer 1971-

Första mikroprocessorn Intel 4004 1971

Första PC Altair 8800 år 1974

Apple 1977

Operativsystemen CP/M, PC-DOS, MS-DOS

Karakteristika: LSI och VLSI ((Very) Large Scale Integration, dvs. tusentals transistorer på en kiselbricka), mikroprocessorer (en hel processor på en enda kiselbricka), strukturerad programmering, applikationsprogram, användarvänlighet, och därmed vidare utbredning

8.9 Femte generationen datorer 1990-

Femte generationens datorer var namnet på japanernas storsatsning 1982-91, ett tioårigt projekt finansierat av Japans handels- och industriministerium. Det hade som mål att utveckla

- användargränssnitt med vanligt naturligt språk, bilder, tal, handskriven text
- automatisk översättning av språk (t.ex. japanska, engelska)
- artficiell intelligens som grund för logikprogrammering och symbolmanipulering
- effektivare teknik och parallellprocessering

Projektet lyckades dock inte så väl att alla mål skulle ha uppnåtts. Projektet uppföljs med ett nytt projekt **Sjätte generationens datorer** (1992-2001) som baserar sig bl.a. på neuralnät.

8.10 Idag ...

Vi lever nu i ett informationssamhälle där en majoritet av människorna sysslar med att behandla data i sitt arbete. Data är en handelsvara.

Internet (”uppfunnet” redan under den tredje generationen datorer) tar över.

Datanät och kommunikation av största betydelse.

Distribuerade system.

Priset går ner och kapaciteten ökar.

8.11 Några anekdoter från historien

Det är lätt att skratta när man hör följande:

1943 lär chefen för IBM, Thomas, J. Watson, ha sagt följande:

- **”I think there is a world market for about five computers”**

1949 hördes följande uttalande:

- **”Where a calculator on the ENIAC is equipped with 18000 vacuum tubes and weighs 30 tons, computers in the future may have only 1000 vacuum tubes and perhaps only weigh 1 1/2 tons.”**

1957 försökte en författare sig på att publicera en bok om databehandling. Men förläggaren svarade:

- **”I have travelled the length and breadth of this country, and have talked with the best people in the business administration. I can assure you on the highest authority that data processing is a fad (ung. modefluga) and won’t last out the year.”**

1977 gjorde chefen för Digital Equipment Corporation (DEC), Ken Olson, följande uttalande:

- **”There is no reason for any individual to have a computer in their home.”**

Chapter 9

Datahantering

9.1 Datahantering

Företag och andra organisationer hanterar stora mängder data.

Varken datorn, programmen eller datahanteringssystem löser alla problem.

Man behöver ett informationssystem som inbegriper program, maskiner och **människan**, dvs. även hur människan inverkar och påverkar data.

Ett företag som säljer varor bör t.ex. kunna behandla uppgifter om följande:

9.2 Datacentrerad och tillämpningscentrerad databehandling

Tillämpningscentrerad datahantering

En lösning på hanteringen av ovanstående data är **tillämpningscentrerad datahantering** (eller dataadministration, sovelluskeskeinen tiedonhallinta): **Varje tillämpning har sitt eget data.**

kunder	fakturering	beskattning
arbetare	kravbrev	anskaffningar
lager	löner	avskrivningar
beställningar	reklam	förfluten tid
leveranser	bokföring	framtid, mm.

Källdata 1 –*longrightarrow* tillämpning 1 –*longrightarrow* resultat 1

Källdata 2 –*longrightarrow* tillämpning 2 –*longrightarrow* resultat 2, osv.

Nackdelar:

- **redundans**: samma data finns på många olika ställen (i många olika filer) vilket leder till slöseri med utrymme
- **uppdateringsproblem**: ändringar i data måste göras på många ställen vilket leder till större kostnader och högre felrisk, t.ex. är det möjligt att motstridiga uppgifter uppkommer
- **databeroende**: datats lagringsform är beroende av tillämpningen. Om tillämpningen förändras måste kanske också lagringsformen ändras
- **datadispersion**: data som är utspridda på olika ställen är svåra att kombinera och de blir ineffektivt utnyttjade. Dessutom är de svårare att skydda mot intrång.

Fördelar:

- små tillämpningar har lätt att komma åt och uppdatera sitt data.
- svårt att komma åt (för obehöriga användare)

Datacentrerad datahantering

Idén med **datacentrerad databehandling** är att hålla ett centrallager, en **databas**, för alla data en organisation behöver.

tillämpning 1 tillämpning 2

/

databas

/ tillämpning 3 tilläpmning 4

En **databas** är en mängd data som beskriver ett visst objekt och som används och uppdateras av ett eller flera datasystem.

Fördelar:

- **mindre redundans**: data lagras en gång på ett ställe i databasen vilket sparar utrymme. Flera olika applikationer kan använda samma data
- **dataoberoende**: Data kan användas på samma sätt även om tillämpningen förändras.
- **dataintegration**: man kan kombinera data lättare och mångsidigare
- **bättre åtkomst**: man kommer åt data utan att man använder någon speciell tillämpning
- **centraliserad säkerhet**: det är lättare att skydda data som befinner sig på ett ställe
- **minskade kostnader**

Nackdelar:

- **komplexitet**: komplicerad programvara som är svår och tidskrävande att konstruera
- **höga grundkostnader** (pga föregående punkt)
- **utbildningskrav**: användarna måste lära sig ett komplext system
- **konversionsproblem**: det kan vara svårt att omvandla ett gammalt system till en databas
- **sårbarhet**: lättare byte för sabotage, stöld, förstörelse, ”alla ägg i samma korg”

Ett **databashanteringssystem** sköter om databasen: åtkomst, uppdatering, skydd, rapportering mm.

9.3 Beskrivning och manipulering av data

Beskrivning

Man skiljer mellan att beskriva data på en **fysisk nivå** , dvs. hur data är lagrade på de olika enheterna (skiva, magnetband, mm.) och på en **logisk nivå**, dvs. hur användare uppfattar att data lagras, t.ex. som namn, adresser, telefonnummer mm.

Den fysiska beskrivningen är viktig t.ex. för operativsystemet som skall läsa eller skriva data. Användaren behöver sällan bekymra sig om den.

Man delar upp databasen i **filer**. En fil är alltså en samling data som har ett semantiskt samband. (En fil kan förstås också vara vilken text som helst eller ett program.)

En fil innehåller **poster** som är en samling dataelement.

En post innehåller **fält**.

Varje fält består av ett antal **tecken**.

Varje tecken kan lagras som en **byte** som innehåller ett antal **bitar** .

Vi har alltså följande hierarki:

databas > fil > post > fält > tecken (byte) > bit.

Exempel:

Anta att vi har en databas som innehåller Helsingfors' telfonkatalog (HTF:s katalog). Vi kan dela upp databasen i tre filer: privatnummer, företagsnummer och gula sidor.

Om vi granskar privatnummerfilen består den av ett antal poster där varje post innehåller fält för efternamn, förnamn, yrke, adress och telefonnummer. Vissa av dessa uppgifter, t.ex. yrke och/eller adress kan saknas. Ett exempel på en post är

Löfström Jonna `läkare Kirurgstigen 52 A 6 504 1532`

Varje fält i posten, t.ex. efternamnet `Löfström`, **s**, **t**, **r**, **ö** och **m**.

Varje tecken representeras av en bitföljd. Enligt ISO Latin1-tabellen representeras t.ex. **L** som 01001100.

För att beskriva dessa uppgifter i en databas använder man ett **datadefinitionsspråk**, ofta förkortat DDL efter den engelska termen. SQL (Structured Query Language) innehåller vissa operationer för att definiera data.

Följande datadefintion med SQL visar vilka logiska data som ingår i en post (tabell) för en telefonabonnent:

```
CREATE TABLE ABONNENTER(
    EFTERNAMN CHAR(20) NOT NULL,
    FORNAMN CHAR (12) NOT NULL,
    YRKE CHAR (10),
    ADRESS CHAR(30),
    TFNNUMMER NUMBER (6,2) NOT NULL
```

Figure 9.1: Exempel på hierarkin vid representation av data i textform.

```
);
\subsubsection{
Manipulering}
För manipulering (uppsökning, utskrivning, förändring,
osv.) av data i en databas krävs ett \concept{datamanipuleringsspråk},
förkortat
DML.

Vanliga operationer i en databas är:

\begin{table}
\begin{tabular}{lll}
Initialisering & Definition av poststruktur\\
               & Ifyllande av poster\\

 Sökning       &Bläddring i databasen\\
               &Frågor\\
               &Tillägg\\
 Uppdatering   &Borttagning\\
               &Förändring av data\\
               &(Förändring av poststruktur)\\
               &Sortering\\
 Arrangemang   &Indexering\\
               &Utskrivning av rapporter\\
\end{tabular}
\end{table}

Exempelvis följande SQL-fråga söker upp och skriver ut
namnen på alla abonnenter som
bor på Kirurgstigen.

\begin{verbatim}
SELECT NAMN
FROM ABONNENTER
WHERE ADRESS LIKE 'Kirurgstigen%' ;
```

SQL-språket innehåller således både definierande och manipulerande operationer. Ett **frågespråk** är ett språk för slå i en databas, dvs. söka data i databasen. I ett frågespråk kan det ingå både ett manipuleringsspråk och ett definitionsspråk.

9.4 Datamodeller

Databasens egenskaper bestäms av hur dataelementen ordnas, hur de kan vara beroende av varandra och hur de behandlas. En d**atamodell** beskriver dessa egen-

skaper.

De vanligaste datamodellerna är

- den **hierarkiska modellen**
- **nätverksmodellen**
- den **relationella modellen** (även relationsmodellen)
- **objektmodellen**

Då en databas följer någon av ovanstående modeller talar man om **hierarkiska datbaser**, **nätverksdatabaser**, **relationella databaser** (**relationsdatabaser**) eller **objektdatabaser**.

Den hierarkiska modellen

I den **hierarkiska modellen** ordnas data hierarkiskt i en trädstruktur.

Den hierarkiska modellen lämpar sig då data till sin natur är trädlikt:

- en far har flera söner (en mor flera döttrar, en förälder flera barn)
- ingen son utan en far

Den hierarkiska modellen är lätt att

- implementera (förverkliga)
- förstå och
- använda

Men den har vissa brister:

- den lämpar sig inte så bra för annat än trädlikt organiserat data
- behovet av mångfaldig lagring uppstår lätt
- en del av data kan vara svåra att komma åt (en son kan endast nås via fadern och frågor i frågespråket blir invecklade)

Figure 9.2: Exempel på en hierarkisk databas.

Figure 9.3: Exempel på en nätverksdatabas.

Nätverksmodellen

I **nätverksmodellen** ordnas data i ett nätverk, dvs data kan ha många kopplingar, flera fäder.

Fördelar med nätverksmodellen:

- allmännare än den hierarkiska modellen (som är ett specialfall av nätverksmodellen)
- smidigare att söka data

Nackdelar:

- komplicerad uppdatering av länkar då data avlägsnas eller adderas

Figure 9.4: Exempel på en relationell databas.

Nätverksmodellen lämpar sig för datahantering då förhållandena mellan posterna sällan förändras.

Nätverksmodellen har sitt ursprung i CODASYL-rapporten från 1971.

Den relationella modellen

Den **relationella modellen** organiserar data i tabeller (relationer) där raderna motsvarar poster och kolumnerna fält.

Förhållandena mellan poster i olika tabeller framgår av de gemensamma fälten.

Tabllerna kan kombineras till nya tabeller (t.ex. totalpris från en kunds beställning)

enligt den **relationella algebran**.

SQL är ett frågespåk för relationella databaser.

Den relationella modellen är begreppsmässigt enkel (människor är vana vid tabeller).

Modellen kan tänkas vara en nätverksmodell där man reserverat länkar för alla tänkbara kombinationer.

Upphovsmannen heter Edgar F. Codd och modellen kom till 1969.

Det tog lång tid för den relationella modellen att slå igenom (kommersiellt) men den är i dag den vanligaste.

Objektmodellen

Objektmodellen delar in data i klasser bestående av objekt. Varje klass har vissa egenskaper (attribut) och vissa metoder (operationer) som får utföras på dess objekt.

Objektmodellen försöker närma sig den verkliga världen

Viktiga egenskaper i objektmodellen

- **inkapsling**: data och dess operationer defineras på ett ställe. Användaren kommer vanligvis åt data endast genom klassens metoder.
- **nedärvning**: klasserna kan ordnas i en hieraki där en underklass ärver egenskaper och metoder av en övre klass.
- **objektidentitet**: varje objekt (i en klass) har sin egen identitet; även om två objekt har exakt samma egenskaper kan de utgöra olika objekt.

Objektmodellen påminner om nätverksmodellen men de viktigaste skillnaderna utgör just nedärvningen, inkapslingen och objektidentiteten.

Objektdatabaser är starkt på kommande.

Objektmodellen används också t.ex. i **objektorienterad programmering** (objektorienterade programspråk såsom C++ och Java), **objektorienterad analys** och **objektorienterad design** (planering).

Figure 9.5: Exempel på en objektdatabas.

Nya trender

I hypertext (Ted Nelson 1965) och hypermedia modelleras data som i nätverksmodellen. Länkar sammanbinder texter (bilder, ljud, film) till ett sammanhängande nät.

Används t.ex. i World Wide Web (webben, WWW).

Fördelar:

- närstående, besläktat data finns på samma ställe, t.ex. man kan länka definitioner till svåra ord i en text.

Nackdelar

- svårt att orientera sig (lost in hyperspace). Man vet inte om man läst allting som berör en viss sak.

Som ett specialfall av nätverkssmodellen (eller den hierarkiska modellen) har man också en modell för **strukturerade dokument**. Ett strukturerat dokument innehåller logiska delar som i sin tur innehåller mindre delar.

Exempel:

En ordbok är ett exempel på ett strukturerat dokument. Ordboken består av ordartiklar som i sin tur är uppdelade i huvudord (lexem), beskrivning av uttal, ordklass, och böjningsformer, samt ett antal moment, ett moment för varje skild betydelse ordet kan ha. Varje moment kan bestå av en beskrivning och ett antal exempel, exemplen i sin tur av meningar, osv.

Det finns olika standarder för att märka ut strukturen i strukturerade dokument: SGML (Standard Generalized Markup Language), ODA (Office document Architecture), HTML (Hypertext Markup Language), osv.

Den viktiga skillnaden till vanliga textbehandlare är att man försöker märka ut den logiska strukturen (se ordboksexemplet ovan), istället för det fysiska utseendet (här skall det vara en större font och fet stil osv.).

Ett strukturerat dokument kan användas som en databas:

- Först måsta man definiera dokumenttypen, dvs. vilka logiska delar dokumentet får innehålla (motsvarar datadefinition)
- Sedan kan man manipulera dokumentet: addera eller ta bort delar, ändra, söka efter vissa delar osv. T.ex. kan man i ordboken söka efter alla ordartiklar som beskriver ett substantiv och som i ett exempel innehåller ordet 'data'.

- Man kan låta skriva ut dokumentet i olika format, på bildskärm, som en webbsida, på papper eller CD-ROM osv.

9.5 Dataskydd

Med olika åtgärder försöker man **skydda data.**

- **reservkopior**: man tar kopior på sina data, t.ex. på en extra diskett
- **reservutrustning**: t.ex. extra strömkällor eller extra maskiner om någon nödvändig enhet slås ut
- **säkra lokaler**: man låser dörren och kättjar fast datorn i bordet/väggen (eller i sig själv).
- **avlyssningsskydd**: datorsystem avger signaler som kan avlyssnas, t.ex. text på skärmen strålar så mycket att den kan "avlyssnas" på 1000 meters avstånd. Ljud från skrivare eller kablar kan också avlyssnas.
- **kryptering**: man omvandlar data så att det är oläsligt. Efter dekryptering går det att använda igen.
- **behörighetsskydd**: användaridentifikation och lösenord
- **nyckelpersonal och arbetsdisciplin**. Alla får inte alla rättigheter på datorsystemet. Det gäller att inte skylta med lösenordet.

Dataskydd är också viktigt för **datasekretessen** (se nedan).

9.6 Datasekretess

Datasekretess, **integritetsskydd** innebär att skydda persondata mot obehörig eller skadlig användning.

Utdrag ur personregisterlagen (471/1987) given 30.4.1987:

- **För att skydda individens personliga integritet samt hans intressen och rättigheter, för att garantera statens säkerhet och för att skapa en god registersed skall vid inhämtande, registrering, användning och utlämnande av personuppgifter iakttas denna lag, om inte annat stadgas genom lag.**

Integritet: rättigheten att själv bestämma om användingen av data som berör en själv

Datasekretessfrågor:

- **VEM** vet **VAD** om **VAD**?
- Kan uppgifterna **KONTROLLERAS** och **KORRIGERAS**?

Problem:

- Riktiga uppgifter missbrukas
- Felaktiga uppgifter används rätt
- Felaktiga uppgifter missbrukas

Exempel på **privata register** är t.ex. kundregister, delägar- och medlemsregister, register som arbetsgivaren har, kreditregister, marknadsföringsregister, mm.

Exempel på **myndigheternas register** är bl.a. det centrala befolkningsregistret, straffregistret, domstolarnas lagfartsregister, utlänningsregistret, centralpolisens register, studentexamensregistret osv.

Några viktiga rättigheter som det stadgas om i personregisterlagen:

- **Insyn i registerbeskrivning**: Var och en skall kunna ta del av registerbeskrivningar (vem håller ett visst register, vem är ansvarig, vad heter registret, varför förs registret och vilka uppgifter innehåller det)
- **Information om kreditupplysningar**: Den som för register över kreditupplysningar måste första gången en person förs in i registret meddela personen om detta
- **Direktreklam**: Adresskällan måste uppges vid direkt marknadsföring
- **Rätt till insyn**: Var och en har i allmänhet rätt till insyn, dvs. man får veta vilka uppgifter som finns om en själv i ett visst register. Avgiftsfritt har man insyn en gång per år.
- **Begäran om insyn**: Man skickar helt enkelt ett brev om att man vill veta vilka uppgifter som finns om en i registret. Om den som för registret vägrar ge ut uppgifterna kan man klaga för **dataombudsmannen**.
- **Begäran om korrigering**: Om det finns fel i ett register kan man kräva att dessa fel rättas till. Uppgifterna kan t.ex vara felaktiga genom att de är oriktiga till sitt innehåll, föråldrade, bristfälliga eller onödiga.

- **Förbud mot utlämnande av uppgifter**: Man kan förbjuda att ens uppgifter lämnas ut för direkt marknadsföring. Inte heller den som för registret får använda uppgifter för annat än vad registret är upprättat för.

Ej rätt till insyn: Register för vetenskaplig forskning, skyddspolisens register, centralpolisens register över tillvägagångssätt, arbetsgivarens manuella kartotek och förteckningar som ligger till grund t.ex. för avlöning. Om registerinsynen kunde skada statens säkerhet eller medföra allvarlig fara för den regisrerades hälsa och vård så kan man också vägras insyn.

För att man ska få upprätta register måste man ha mycket goda skäl. Endast i vissa undantagsfall får man registrera **känsliga uppgifter** såsom

- upgifter som beskriver ras eller etniskt ursprung,
- samhällelig eller politisk uppfattning,
- religiös övertygelse,
- sexuellt beteende,
- hälsotillstånd och handikapp,
- brottsliga gärningar samt
- socialvårdstjänster som den registrerade erhållit.

Lagen begränsar också rätten till **samkörning** av personregister, dvs. att man kombinerar olika register. På det sättet kunde man få fram en helhetsbild av en viss person.

Datasekretessmyndigheterna i Finland består av **dataombudsmannen** och **datasekretessnämnden**.

Nuvarande dataombudsman är avd. chef Reijo Aarnio 1.11.1997- (tidigare var det vicehäradshövding Jorma Kuopus 1.11.992-31.10.1997). Dataombudsmannen [se `http://www.tietosuoja.fi/ruotsi.html`]

- handlägger och beslutar i frågor som gäller personregister på det sätt som stadgas i personregisterlagen (471/87 ändr. 1011/89, 387/94 och 630/95)
- utövar tillsyn över samt styr och kontrollerar inhämtandet, registreringen, användningen och utlämnandet av personuppgifter så att syftena med personregisterlagen nås
- följer med den allmänna utvecklingen av personregisterpraxis och tar sådana initiativ som han finner nödvändiga

- sörjer för att information lämnas om sektorn
- sörjer för det internationella samarbete som gäller personregister

Datasekretessnämnden

- behandlar frågor som är av principiell vikt för tillämpningen av personregisterlagen
- kan ge tillstånd till undantag från stadgandena i personregisterlagen och personregisterförordningen
- beslutar om tillståndsansökningar som gäller utlämnande av personuppgifter utomlands
- kan på ansökan av dataombudsmannen ålägga den registeransvarige att rätta till sådant som har gjorts orättmätigt eller fatta beslut om förbud i saken
- kan föreskriva att registerverksamhet skall upphöra
- följer med behovet att utveckla lagstiftningen om personregister och tar sådana initiativ som den finner nödvändiga

Dataombudsmannens byrå och datasekretessnämndens sekreterare arbetar i Helsingfors. Adressen är Albertsgatan 25, 3 vån. Byrån är öppen under tjänstetid klockan 8.00-16.15.

Chapter 10

Datakommunikation

Datakommunikation innebär överföring av data på längre avstånd.

Sändare och mottagare måste vara överens om hur de kommunicerar. Detta definieras i **protokoll**. Ett protokoll berättar bl.a.

- vem som får överföra data och när
- hur data överförs
- hur fel uppfattas och behandlas
- hur meddelanden tolkas

En traditionell terminal byggs för ett visst protokoll. Nyare maskiner, t.ex. persondatorer som inte egentligen är terminaler, **emulerar** ett visst terminalprotokoll så att man kan använda dem som terminaler. Emulering innebär att man låter datasystemet efterlikna ett annat; simulering igen innebär att man representerar egenskaper hos ett reellt eller abstrakt system genom egenskaper hos ett annat system.

Antalet olika protokoll är stort. Vi koncentrerar oss på några parametrar som ingår: överföringsriktning, överföringsmetod, överföringshastighet och överföringsfel.

10.1 Några begrepp

överföringsriktning

Simplexöverföring innebär att överföring bara kan ske i en riktning. Exempel: Från ett mätinstrument till datorn.

Halvduplex överföring innebär att data kan sändas bara i en riktning i gången.

Full duplex innebär att data kan sändas i båda riktningarna samtidigt.

överföringsmetod

över långsamma linjer sker överföringen **asynkront** vilket betyder att mottagaren inte vet när data kommer. Därför måste varje tecken inledas och avslutas med speciella signaler.

Synkron överföring innebär att sändare och mottagare har "synkroniserat" sina klockor. Mottagaren vet när data sänds och är beredd. Data sammanförs i större block som inleds och avslutas med speciella signaler.

Seriell (sekventiell) **överföring** innebär att data överförs en bit i gången i en ledning.

Parallel överföring innebär att data överförs parallellt, t.ex. ett teckens bitar i parallella ledningar samtidigt. Parallell överföring används speciellt inom datorn.

överföringshastighet

Hastigheten mäts i bitar per sekund, förkortat **bps** (bits per second).

Överföringshastigheterna varierar stort. De långsammaste modemen har en hastighet på kanske 75 bps, de snabbaste fiberoptiska datanäten 1 Gbps (10^{12} bps).

Vanliga hastigheter är t.ex. 9600 bps, 14400 bps, 19200 bps, 38400 bps.

överföringsfel

Det finns alltid risk att data förvrängs av olika störningar.

Det kan hända t.ex. att 1-bitar överförs som nollor eller vice versa.

För att bekräfta att en signal är korrekt bifogas kontrollinformation i signalen, s.k. **redundanskontroll**.

Exempel på redundanskontroller:

- **paritetskontroll**: ett tecken kompletteras med en extra bit. Vid udda paritetskontroll bör alla kompletterade tecken innehålla ett udda antal ettor; vid jämn

paritet ett jämnt antal ettor. Om mottagaren mottar ett tecken som inte följer rätt princip (har udda resp. jämn paritet) har den funnit ett fel och kan be om tecknet på ntt.

- **validitetskontroll**: man utnyttjar kunskapen om att alla bitkombinationer inte är möjliga. Man kontrollerar då att den överförda koden är någon av de tillåtna eller kända.
- **användning av kontrollsummor**: Man betraktar alla tecken som ett tal mellan 0 och 255 och adderar alla tecken i ett block. Denna kontrollsumma jämförs sedan av mottagaren.

Standardisering

Datakommunikation har standardiserats i **OSI-modellen** (Open Systems Interconnections). OSI-modellen delar upp kommunikationen i skikt eller nivåer (7 stycken) där varje skikt fokuserar på ett kommunikationsproblem. Nivåerna är det fysiska skiktet, länksskiktet, nätskiktet, transportskiktet, sessionsskiktet, presentationsskiktet och applikationsskiktet.

10.2 Historik

Datakommunikationen har utvecklat sig vid sidan om ADB:n. Således har vi internationella uppfinningar såsom

- den optiska telegrafen 1791 i Franktrike
- eltelegrafen 1812 i Tyskland
- telefonen 1876 i USA
- telex 1877 i Frankrike

och i Finland

- den optiska telegrafen Finland-Sverige 1802
- eltelegrafen Helsingfors - St. Petersburg 1855
- den första telefonlinjen i Forssa 1878
- Helsingfors telefonförening 1882
- dataöverföring i telefonnätet 1964 (Kesko Oy)

10.3 Datakommunikationssystem

Datakommunikationssystem kan bestå av:

- terminaler:
  - en terminal är inte bara en bildskärm med tangentbord, det kan också vara en skrivare
  - en terminalemulator gör att t.ex. en mikrodator härmar en riktig terminal
- gränssnittsenheter: där hårdvaran sköter om att data överförs till/från datakommunikationslinjerna (t.ex. modem)
- datakommunikationslinjer
  - kabel (telefonlinje, koaxialkabel, optisk fiber)
  - radiosignaler (även över kommunikationssatelliter)
- datorer; en speciell dator kan fungera som värddator som styr nätets funktioner

10.4 Datanät

Ett **datornät** består av flera datorer kopplade samman med varandra.

Genom att koppla upp sig på nätet kan man nå vilken dator som helst.

Ett datornät där datorerna är nära sammankopplade kallas för **distribuerat system**.

Datanätet sammanbinder datorerna i ett datornät.

I ett **lokalt nät** ligger datorerna inte långt ifrån varandra (kanske högst någon kilometer). Motsats **icke-lokalt nät**.

Datorerna i ett nät kan vara kopplade på olika sätt. Strukturen kallas **topologi**. I ett **bussnät** är alla datorer kopplade till en central kommunikationsväg. I ett **slingnät** (ringnät) bildar datorerna och datanätet en ring. Och i ett **stjärnnät** är alla utom en dator anslutna till en central enhet.

Figure 10.1: Universitetes lokala nät.

Exempel på nät

Telefonnätet är omfattande och kan användas för datakommunikation.

Telefonnätet är analogt (gjort för tal, dvs. ljud) och digitala (dvs. alla) data måste därför omvandlas till analoga dito med ett **modem** innan de kan skickas i väg. På samma sätt måste ankommande analoga data omvandlas tillbaka till digitala data när mottagande dator skall läsa dem. Modem kan ha hastigheter som t.ex. 14000 bps eller 28800 bps.

Radiotelefonnätet (NMT, GSM).

Telex (Finland 1945-) med vilket man kan skicka "telegram" mellan företag. Detta är dock långsamt (och gammalmodigt).

Teletex (Finland 1983-) som en mera utvecklad form av telex.

Det **allmänna datanätet** omfattar digitala förbindelser över hela Finland.

Det finns också **privata nät** (företag mm.).

Det består av mindre lokala nät och mikronät. Förbindelserna går vidare till FU-

NET (ett datanät för universitet, högskolor och forskningsanstalter i Finland), NORDUNET (motsvarande för Norden) och Internet (motsvarande för världen). Idag har också många företag, skolor, etc., kontakt med internet.

På det här sättet kan vi köra upp på **datamotorvägen** .

10.5 Internet

Ursprung på 1960-talet i ARPANET, senare anslöt sig många amerikanska universitet. Internet spred sig till Europa och senare över hela världen.

Internet är alltså ett datanät eller i själva verket ett stort antal (¿ 500 000) sammankopplade datanät.

Idag ca 20 miljoner datorer anslutna (7/97), och tiotals miljoner användare (hundra miljoner?).

Internet erbjuder förbindelse mellan datorer. På nätet kan man sedan använda olika serviceformer, bl.a.:

- **elektronisk post**:
  - Man kan snabbt och förmånligt skicka brev åt en eller flera mottagare, som inte måste finnas på plats då meddelandet kommer fram (jämför telefon) utan kan läsa det när de hinner.
  - För att ta emot (och skicka meddelanden) behöver man en internetadress som i stort sett ser ut såhär: `användarid@maskin.organisation.land`, t.ex. `linden@cs.helsinki.fi`.
  - Elektronisk post används inte bara för brev till enskilda personer utan också för utdelning till flera på en gång, t.ex. i postningslistor (en grupp som skriver sinsemellan till varandra om något ämne) eller för att skicka ut elektroniska tidningar.
  - Elektronisk post kan också användas till sökning i databaser mm.
  - även marknadsförare har upptäckt e-posten och det kommer mycket skräppost, s.k. **spams**, över nätet.
- **diskussionsgrupper**:
  - En sorts elektronisk anslagstavla där man kan läsa meddelanden och även skicka in meddelanden själv.
  - Varje grupp har sitt eget ämnesområde och det finns tusentals olika grupper.

  - Exempel på diskussionsgrupper är: `hy.tktl.tiedotukset`, `rec.music.opera`, etc.

- **samtal i realtid - IRC** (Internet Relay Chat):
  - Flera personer kan "samtala" (skriva meddelanden åt) med varandra i realtid.
  - Alla samtalare ser samma text som dyker upp på deras skärnm, dvs, alla "inlägg" som görs.

- **webben** (World Wide Web):
  - Webben baserar sig på hypermedia, dvs. dokument med text, bilder, ljud, mm och länkar mellan dessa dokument.
  - För att kunna "surfa" på webben (dvs söka upp dokument med hjälp av internetanslutningen från andra datorer) behöver man en **webbläddrare**, t.ex. Netscape, Lynx eller Microsoft Explorer.
  - Webbdokument är uppgjorda med ett språk som kallas HTML (Hypertext Markup Language) och överföring sker mestadels med HTTP (Hypertext Transfer Protocol).

- **filöverföring** FTP (File Transfer Protocol):
  - man kan föra över filer (av alla slag: texter, program, bilder) genom FTP. FTP kan även användas i en webbläddrare. (En annan möjlighet HTTP).

- **terminalkontakt** (telnet):
  - man kan logga in på en maskin som ligger långt borta. Då måste man naturligtvis också ha ett användarid och ett lösenord på den maskinen.

- **gopher**:
  - en sorts elektronisk anslagstavla där man kan plocka hem filer eller söka i databaser.

10.6 Några termer som rör datakommunikation

- **ATM**: Asynchronous transfer mode - ett nytt system för snabbare överföring, bl.a. av levande bilder (film).
- **GSM**: Global System for Mobile Communications - ett digitalt nät för mobiltelefoner.
- **NMT**: Nordisk Mobil Telefon - ett vanligt nät i Norden som fungerar nmed frekvenserna 450 MHz och 900 MHz.

- **TCP/IP**: Transmission Control Protocol/Internet Protocol - ett protokoll som används på internet.

10.7 I framtiden

Vad kan vi vänta oss i (en nära) framtiden? Redan nu har vi möjlighet till

- HDTV: High Definition Television - digital TV, bättre resolution (avlång ruta)
- interaktiv TV - tittaren påverkar programmen
- video on demand (video på begäran, beställning). TV-tittaren knappar in vilken film han/hon vill se; videon överförs på datanätet på några sekunder
- elektroniska kontanter och att göra uppköp på nätet

Chapter 11

Systemutveckling

En organisation köper en dator. Men ingenting blir bättre än förr bara man ”trycker på en knapp”.

En fördjupad studie av organisationens funktioner behövs.

Problemen kan vara svåra att hitta och lösningarna likaså. Förändringar i tidigare rutiner kanske krävs.

Man behöver systemplanerare som utför systemanalys och -planering för att förbättra det nuvarande systemet eller för att bygga upp ett nytt datasystem.

Ett **datasystem** är ett system som betjänar någon funktion genom att behandla data. Med ett datasystem avses vanligen en databehandlingshelhet bestående av människor samt databehandlings- och transportenheter. Men det kan också uppfattas som ett abstrakt system som består av data och dess behandlingsregler.

Systemanalys innebär kartläggning och värdering av nuvarande funktioner. Samtidigt försöker man se om funktionerna motsvarar målen.

Systemplanering innebär att man planerar det nya systemets uppbyggnad utgående från systemanalysen.

Systemplaneraren sysslar både med systemanalays och -planering.

Det finns många fler yrkeskategorier: **ADB-planeraren** sköter teknisk planering av program och val av apparatur. En **programmerare** skriver dataprogrammen och testar dem och en **operatör** sköter datorerna.

Systemarbete inbegriper alltså undersökning, planering och implementering (förverkligande).

För byggande av ett system startar man ett **projekt**. Ett projekt kan ha följande

skeden

1. Förundersökning
2. Systemanalys
3. Systemplanering
4. Implementering av systemet
5. Ibruktagning och underhåll av systemet

Arbetet framskrider inte alltid i ordningsföljd, utan man måste vanligen återvända till tidigare skeden (oftast bara till föregående).

Förundersökning

I **förundersökningen** frågar man sig om man överhuvudtaget behöver göra något alls.

Utgångspunkten är att problem har upptäckts:

- Vad är problemet? - Ledning, systemplanerare och användare bör uppnå enighet om problemet.
- Vad kostar det att lösa problemet och hur lång tid tar det (storleksordning) - Inga beslut kan fattas utan kostnads - och dylika beräkningar
- Presentation av lösningsalternativen i huvuddrag - Ska man ändra ett gammalt system eller bygga ett helt nytt? Ett hurdant system behöver man och hur ska det byggas?
- Rapport till beslutsfattarna - Vilka för- och nackdelar finns? Beslutet kan vara att man inte gör någonting. Om man besluter sig för att göra någonting kan rapporten visa riktningen för åtgärderna.

Systemanalys

Utgångspunkt: Projektet fortsätter i den riktning förundersökningen anger.

Vad/hur gör man nu?

Vad/hur tycker användarna och ledningen att det vore ändamålsenligt att göra?

- Insamling av data
  - Skriftliga dokument
  - Skriftliga förfrågningar
  - Intervjuer
  - Observationer
- Dataanalys
  - Dataflöden
    - Varifrån och vart
    - Lagring av data
    - Behandling av data
    - Rutter
  - Dataindex
    - Datas namn, beskrivning, källa, form, användningsändamål
- Svarta lådor - ger en specifik funktion, vad men inte hur

Systemplanering

Utgångspunkt: Man vet vad som behöver göras och planerar hur det skall göras.

- Genomgång och precisering av projektets målsättning
- Planering av systemets funktion - skriftligt in i minsta detalj
  - Utmatning
  - Inmatning
  - Datalager
  - Databehandling
  - Skydd och kontroller
  - Personalens åtgärder
- Sammandrag för beslutsfattarna
  - Funktionen hittills och dess problem
  - Fördelar med det planerade systemet
  - Allmän struktur för det planerade systemet
  - Kostnadskalkyl för implementeringen - speciellt förändringar jämfört med tidigare
  - Tidtabell

Implementering av systemet

Utgångspunkt: Man vet vad som skall göras och hur det skall göras. Man gör det!

- Tidtabell för implementeringen
- Programmering enligt detaljerade anvisningar
- Grundlig testning
- Dokumentering, bl.a. anvisningar åt användarna

Ibruktagning och underhåll

Utgångspunkt: Systemet är färdigt. Det bör fås att fungera i produktionsanvändning

- Omvandling av filer
  - Indata
  - Datalager
- Skolning
  - Användarna bör kunna använda det nya systemet
- Byte av system
  - Allt på en gång? En bit i sänder? Prova sig fram?
  - Det gamla och det nya parallellt en viss tid
- Uppföljning
  - Otillfredställande punkter rättas
- Underhåll
  - Fel hittas
  - Omgivningen förändras
    - Avlägsning, tillägg, förändringar
  - Nya versioner tills det blir aktuellt att börja om från början

Chapter 12

Avslutning

Detta kapitel innehåller kort och gott allt som blev över eller saker som jag glömde. Vi tar upp bl.a.

- specialområden in om datavetenskapen: artificiell intelligens, mm.
- säkerhet och sårbarhet (databrott)
- hälsa och ergonomi
- vad framtiden för med sig

12.1 Artificiell intelligens

Turings test

Artificiell intelligens (AI) strävar till att datorerna skall kunna härma mänskligt beteende (inlärning, beslutsfattning).

När är en maskin intelligent?

Turings test:

En person ställer frågor till en maskin och till en annan människa som båda är dolda för frågeställaren. Om frågeställaren inte kan avgöra när det är maskinen och när det är människan som svarar, visar maskinen prov på intelligens. Dvs. om sannolikheten för att gissa rätt är 50

På något begränsat område kan en maskin tyckas intelligent men i det allmänna fallet ...

Delområden inom artificiell intelligens

AI omfattar bl.a.

- expertsystem
- kunskapsbaserade system
- regelbaserade system
- maskininlärning
- behandling av naturligt språk även i form av talad text (natural language processing, NLP)
- bildbehandling
- mönsterigenkänning
- neuralnät
- intelligenta agenter

Expertsystem är programvara som inom ett begränsat område uppvisar problemlösningsförmåga jämförbar med mänskliga experter på området.

Kunskapsbaserade system är programvara som baserar sig på en i systemet explicit lagrad kunskapsbas.

Regelbaserade system är programvara som implementeringstekniskt är utformad med en formalism av regler.

Neuralnät består av mängder enkla beräkningselement kopplade till varandra (idé: hjärncellerna i hjärnan).

Exempel på applikationer inom AI:

- robotar som ser (registrerar sin omgivning)
- schackspelande datorer
- bagagekontroll på flygplatser (neuralnät)
- ...

Våren 1997 hände det första gången att ett datorschackspel vann över en världsberömd schackspelare. Då slog IBMs Deep Blue program ut världsmästaren Garri Kasparov. Ett annat exempel på artificiell intelligens stod NASAs Pathfinderrobot för;

den anlände sommaren 1997 till Mars för att köra omkring där och ta prover. För en sådan robot är det viktigt att den både kan se sin omgivning och med hjälp av detta t.ex. undivka hinder i vägen.

Expertsystem

Ett **expertsystem** innehåller en samling fakta och en lista regler för att dra slutledningar av fakta. Expertsystemet frågar användaren då tilläggsfakta behövs.

Expertsystemet har ofta en förklaringsmekanism som berättar hur det kommit fram till sin slutledning (självförklarande programvara).

Expertsystem är ofta skrivna på Lisp, Prolog eller motsvarande språk.

En **kunskapsingenjör** (dvs. AI-programmerare) överför kunskap från en (möjligen ADB-okunnig) expert inom något särskilt område till expertsystemets fakta- och regeldatabas (= typ av kunskapsbas).

Exempel på expertsystem:

- Mycin - diagnos av infektionssjukdomar
- Prospector - evaluering av mineralfyndigehter
- XCON - sammansättning av apparatur till datorsystem
- Delta - service av diesellok
- även inom försäkringsbranschen, kreditvärdighet, skattedeklaration, mm.

Exempel på ett simpelt expertsystem: Ett system som gissar vilket djur vi tänker på genom att ställa frågor till användaren.

Från början har vi lagt in fakta om t.ex. tio däggdjur, tio fågelarter, tio fiskarter och tio insektarter. Varje djurart är noga beskriven, t.ex. en elefant är beskriven på följande vis: däggdjur, mycket stor, har värdefulla betar, finns i Afrika och Asien, blir mycket gammal, har snabel, har grå tjock hud, osv.

Vi ber nu datorn gissa vilket djur vi tänker på. Antag att vi tänker på just en elefant. Dialogen med datorn kan bli på följande sätt:

```
Datorn: Är det en insekt?

Vi: Nej.
```

```
Datorn: Är det ett däggdjur?

Vi: Ja.

Datorn: Är det mycket stort?

Vi: Ja.

Datorn: Är det en flodhäst?

Vi: Nej.

Datorn: Är det en elefant?

Vi: Ja.
```

På det sättet ringar in datorn det djur som vi tänker på genom att hela tiden jämföra våra svar med vad programmet redan vet. Antag att djuret som vi tänker på är en noshörning som inte finns med i programmet.

```
Datorn: Är det en elefant?

Vi: Nej.
```

Eftersom datorn inte hittar något alternativ, ber den användaren att lära den detta djur.

```
Datorn: Beskriv djuret!
```

Vi beskriver noshörningen med relevanta egenskaper och beskrivningen förs in i kunskapsbasen. Nästa gång vi tänker på en noshörning kommer datorn att veta svaret. (Obs. Jämför med leken där en person tänker på något och alla andra får ställa frågor, men bara sådana frågor som den första personen kan svara ja eller nej på. Den som först gissar det rätta svaret vinner. I leken blir de tävlande tvungna att använda sina egna (nästan oändliga) kunskapsbaser om världen.)

På motsvarande sätt kan datorn ställa frågor om patientdata, dvs. observationer som t.ex. temperatur, blodtryck, mm. och sedan med hjälp av sin kunskapsbas dra slutsatser om vilken sjukdom patienten har. Ett sådant system kräver dock alltid att en expert, en läkare, tar ansvar för beslutet i sista hand.

Intelligenta agenter

En **intelligent agent** är ett system (hårdvara eller mjukvara) som är självständigt, interaktivt och reaktivt mot sin omgivning och mot andra agenter.

En intelligent agent kan vara ett program som lär användaren knep och knåp inne i ett annat program. Ett exempel är den nya versionen av Word som har en sådan agent. Agenten följer med vad användaren gör i textbehandlaren, vilka kommandon han använder, vilka upprepningar han gör och ger sedan förslag på hur användaren kunde ha gjort det bättre/snabbare/effektivare.

Ett annat exempel är en **personlig agent** (alltså ett dataprogram) som surfar på nätet och söker information som intresserar en viss användare. Användaren ger vissa termer, t.ex. datavirus och databrott. Agenten söker sig sedan självständigt fram på nätet efter webbsidor med dessa termer. När arbetsdagen är slut kanske agenten dyker upp på skärmen och berättar vad den hittat.

12.2 Säkerhet och sårbarhet

Vi har tidigare behandlat dataskydd (hur man skyddar data/datorer mot obehörig användning) och datasektretess (vilka regler som gäller persondata).

Här ska vi titta på vilka databrott man kan utsättas för (om man kanske inte skyddat sig tillräckligt).

Databrott

Människor som begår brott är som bekant ingenting nytt i historien.

Datorer ger nya möjligheter till brott som dessutom kan vara svåra upptäcka. De brott som avslöjas utgör endast en bråkdel av alla begångna brott (uppskattning).

Många som drabbas av databrott låter bli att anmäla det för att undvika negativ publicitet. Tänk t.ex. på en bank som har så dåligt säkerhetskydd att någon lyckats med förskingring. Vem vill ha sina pengar där?

Tidigare brottslingar anställs också ibland som datasäkerhetschefer på samma bolag!

Olika typer av databrott:

Programmanipulationer En programmerare gör en ändring i ett program i syfte att lura sitt företag, t.ex några extra instruktioer i ett program för utbetalningar; han sätter in en s.k. **trojansk häst**. En trojansk häst kan "spricka" vid en viss inställd tidpunkt, dvs. instruktionerna körs i gång på ett visst datum och ett visst klockslag. Den blir då en **tidsbomb**.

Det klassiska exemplet är en programmerare på en bank i New York som i början av 1970-talet via ett antal falska instruktioner tog hand om alla cents-avrundningar som uppstod i bankens transaktioner och samlade dessa på eget konto.

En anna programmerare i Hamburg förde över en summa motsvarande ca 3 penni från alla kunders konton. Vem bryr sig om det fattas 3 penni. Detta kallas s**alamimetoden**: man skär tunna skivor från flera ställen; dvs. många bäckar små ...

En **lönndörr** innebär att en programmare sätter in ett eget lösenord med vilket han senare kan ta sig in i programmaet för att göra databrott.

Användarbrott En annan form av manipulation är användarbrott där brottslingen manipulerar med de dagliga rutinerna.

Ett exempel är en kvinna på ett företag i Göteborg som lade in påhittade leverantörer i reskontraprogrammen för att kunna beordra utbetalning till dessa, dvs. sig själv.

Stöld av program och data Program är dyra och det kan löna sig att kopiera ... tills man åker fast.

Ett exempel är Karolinska institutet som misstänks ha gjort s.k. piratkopior av program.

Det kan också vara ett brott att använda företagets datorer för att utveckla program som man säljer för egen räkning.

Obehörigt intrång Ett databrott är att otillåtet ta sig in på något datasystem, t.ex. genom att testa olika behörighetskoder tills man träffar rätt. En sådan brottsling kallas **cracker** (systembrytare). Och brottet kan naturligtvis vara mera eller mindre gravt.

Ibland ser man också termen **hacker** för en person som tar sig in i ett system men hacker kan också avse en som håller på och "hackar" på sitt tangentbord i all oändlighet, dvs. är en datafreak i största allmänhet. En hacker ser vanligen ner på crackers.

Avlyssning överföring av data på telenätet kan kopieras.

En dataskärm kan avlyssnas med pejlingsutrustning.

Datavirus Ett **datavirus** är en variant av programmanipulation, dvs. ett antal instruktioner som förs in i en dator och lurar denna att göra fel saker.

Datavirus kallas så för att de förutom att de får program att tappa kontrollen ofta också kan föröka sig och sprida sig, kanske främst via disketter eller nerladdade program (via internet).

Det finns en otrolig mängd olika virus.

Ett virus kan t.ex. vara ganska oskyldigt och bara visa någon (oanständig eller irriterande) figur på rutan vid vissa tidpunkter. Ett virus kan också vara synnerligen "malignt" och t.ex. förstöra filer för användare eller ta över systemets kapacitet.

Ett **virusbekämpningsprogram** känner till ett visst antal virus och kan alarmera användaren och möjligen förstöra viruset. Men så kommer det ett nytt virus som bekämpningsprogrammet inte känner till datapersonal och användare världen över hinner oroa sig någradagar innan man finner ett nytt "vaccin".

Man kan försöka undvika smitta genom att vara noggrann med vilka disketter man använder var och när, och vilka program man laddar ner över nätet (om man alls gör det); samt genom att regelbundet (t.ex. varje gång man loggar in) köra ett virusbekämpningsprogram som kontrollerar såväl hårdskiva som disketter.

Sårbarhet

Det finns en viss sårbarhet i samhället på grund av databrott, otillräckligt dataskydd eller datasekretess.

Sårbarheten har också sin grund i

- **koncentrationen** av data till stora system och beroendet mellan dessa. Om system eller förbindelser slås ut ligger vi illa till. å andra sidan kom internet delvis till för att klara av sådana situationer. Internet bara kan inte slås ut; om en förbindelse bryts, hittar data alltid en annan väg.

- **personalberoendet** i företag och myndigheter där ett fåtal personer klarar av de komplexa systemen och där dessa personer i själva verket får väldigt mycket makt eller blir oumbärliga för företaget. Vad händer sedan om dessa personer utsätts för utpressning?

- **känsligt innehåll** i många register. Om man missbrkar dessa register kan det ha katastrofala följder. Register över människor, företag, vägnät, fastigheter mm. kan utgöra ett allvarligt hot mot ett land vid t.ex. en krigssituation.
- **utlandsberoendet** där den mesta tillverkningen sker utomlands (utanför Finland) vad gäller datorer, reservdelar, och även delvis program. Främmande makt kan t.ex. utöva påtryckning.

12.3 Ergonomi

Allt fler arbetar med data.

Det är viktigt att både arbetsmetodik och arbetsmiljö (= ergonomi) är anpassad för människan.

Redan nu känner man till problem som uppkommer vid dataarbete såsom

- ont i arm och handled, s.k. musarm
- ont i nacke och rygg
- problem med synen
- pyskiska problem
- dataskärmsstrålning
- elallergi

ögonbesvär

För att undvika ögonbesvär som grus eller trötthet i ögonen är det viktigt att

- dataskärmen är reflexfri och vridbar
- texten är tydlig och tillräckligt stor
- ljusförhållandena i rummet är lämpliga

Man kan t.ex. skaffa sig specialglasögon som man (bara) använder när man arbetar vid en skärm.

Allt detta hjälper till att skapa en god **synergonomi**.

Rygg- och nackbesvär

För att undvika rygg- och nackbesvär ska man se till att

- man har en bra stol där ryggstöd och sitthöjd går att reglera
- bordet går att reglera
- man tar pauser i arbetet och rör på sig, t.o.m. gymnastiserar med jämna mellanrum

Med fel belastning på arm och hand när man använder musen kan leda til en s.k. **musarm**. En mus är lätt att använda även fysiskt, men om man i fel ställning måste göra tusentals klickningar kan det sluta illa.

Strålande skärmar

En dataskärm sänder ut elektromagnetisk strålning. Dessutom bildas det ett elektrostatiskt fält kring skärmen.

Man brukar rekommendera att gravida kvinnor inte skall arbeta vid dataskärm men forskarna är oeniga om hur farligt det kan vara.

Nya dataskärmar strålar dock mindre än förr.

Dataskärmar med flytande kristaller eller plasmaskärmar varken strålar eller har elektrostatiska fält.

Vissa personer lider av **elallergi** vilket gör att de kanske inte alls kan använda elektroniska apparater. även här är forskare oense om orsaker och problem.

Psykisk arbetsmiljö

Problem som uppstår med den psykiska arbetsmiljön kan vara t.ex. stress som grundar sig på

- driftavbrott eller det att arbete samlas på hög vid driftavbrott.
- orolighet för säkerhet pga. av strålning och elektrostatiskt fält.
- att man är rädd för att datorn tar över ens jobb eller att man förlorar inflytande att bestämma hur jobbet skall göras.

- brist på kunskap och utbildning, man känner sig dum, otillräcklig och undanskuffad.
- vetskap om att datorn används för att kontrollera vad man gör och när. Man kan t.ex. kontrollera när en person börjar jobba (försenat?), hur mycket den jobbar osv.
- hur stor del av arbetstiden man sitter vid datorn. Om man har kvalificerade uppgifter och tillbringar mindre än hälften av sin tid vid en terminal ser man den oftast bara som ett effektivt hjälpmedel i arbetet.

Man kan försöka förbättra den psykiska arbetsmiljön på olika sätt t.ex. genom att låta arbetsuppgifter rotera, utbilda personalen mm.

12.4 I framtiden

Utvecklingen har varit snabb inom ADB:n.

1943 lär chefen för IBM, Thomas, J. Watson, ha sagt följande:

"I think there is a world market for about five computers"

1977 gjorde chefen för Digital Equipment Corporation (DEC), Ken Olson, följande uttalande:

"There is no reason for any individual to have a computer in their home."

Det som hände på 60-, 70- och 80-talen är gammal historia. Samma sak gäller delvis det som hände ifjol!

Vissa tendenser kan man skönja

- internetanvändningen ökar - talförbindelse, nätradio, visuell telefon, video-on-demand, kommersialisering, nätpengar och -uppköp, försök till censur
- artficiell intelligens dyker upp inom flera olika områden
- fjärrarbete som betyder att man arbetar där man vill, t.ex. hemma eller på Rivieran
- virtuella världar innebär konst och underhållning såväl som naturtrogen simulering

Men det är omöjligt, kanske t.o.m. dumt och oförsiktigt att säga någonting definitivt om det som händer om några år.

Chapter 13

Referenser och bredvidläsning

- Brookshear, J. Glenn: **Computer Science: An Overview**. Addison-Wesley, 1997.
- Datasekretess - en medborgerlig rättighet, en broschyr från dataombudsmannens byrå.
- Dataombudsmannens webbsidor: `http://www.tietosuoja.fi/ruotsi.html`
- Floréen, Patrik: Introduktion till datateknik: Föreläsningsunderlag hösten 1994. Kompendium D350.
- Lunell, Hans: **Datalogi - begreppen och tekniken**. Studentlitteratur, 1994.
- Pekka Orponen: **Ensimmäisen tietokoneen loikin Turing** i Tietotekniikka 2, 1987, sid. 43-46.
- Slotnick et al. Computers & Applications. D.C. Heath and Company, 1989.
- James E. Tomyako. Artikeln **Anecdotes** i Annals of the History of Computing, vol. 11, nr 4, 1989.
- Westberg, Per: Praktisk ADB, Studentlitteratur, 1989.
- Wikla, Arto: Tietotekniikan alkeet: Luentomateriaali 1996. Kompendium D376.

Chapter 14

Introduktion till datateknik - Ordlista

Varning! Om en dataterm inte förekommer i ordlistan, betyder det inte att den är oviktig. Föreläsaren har kanske bara råkat missa den. Ordlistan kan också innehålla direkta fel. Om du upptäcker ett sådant meddela föreläsaren.

Ordlistan är uppdelad enligt avsnitten på kursen. Därför förekommer några ord också flera gånger då termen behandlats i olika avsnitt.

14.1 Databehandling blir allt viktigare

databehandling, informationsbehandling	tietojenkäsittely =	data processing, information processing
automatisk databehandling, ADB	automaattinen tietojenkäsittely, ATK	electronic data processing, EDP
manuell databehandling, MDB	”käsin tapahtuva tietojenkäsittely”	manual data processing
dator, datamaskin	tietokone	computer
automation	automaatio	automation
datateknik information technology	tietotekniikka datalogi, datavetenskap	informatics, informati computer scien
	tietojenkäsittelytiede	
algoritm	algoritmi	algorithm
programvara	ohjelmisto	software
maskinvara	laitteisto	hardware
kulram	helmitaulu	abacus

räknesticka	laskutikku	slide rule

14.2 Data och datarepresentation

data, pl.b.f -na el. sg. b.f. -t	tieto	data
information	informaatio	information
tolkning av data	tiedon tulkinta	interpretation of data
datarepresentation	tiedon esitys	data representation
analog representation	analoginen esitys	analog representation
digital representation	digitaalinen esitys	digital representation
diskreta data	diskreetti tieto	discrete data
binära (tal)systemet	binääri(luku)järjestelmä	binary (numeration) system
binär siffra, bit	binääri numero, bitti	binary digit, bit
ord	sana	word
decimalsystemet tiosystemet	kymmenjärjestelmä	decimal number system
oktalsystemet, åttasystemet	kahdeksanjärjestelmä	octal number system
hexadecimalsystemet, sextonsystemet	kuusitoistajärjestelmä	hexadecimal number system
konvertering	muunnos, muuntaminen	conversion
text	teksti	text
teckenkod	merkkikoodi	character code
tecken	merkki	character
teckensträng	merkkijono	(character) string
tal	luku	number
heltal	kokonaisluku	integer
bråk(tal)	murtoluku	fraction
flyttal	liukuluku	float point number
karakteristika		
mantissa		
logiskt värde	looginen arvo	logical value
ljud	ääni	sound
bild	kuva	image
raster-, pixelgrafik, rasterdatagrafi	rasterigrafiikka	raster graphics
vektorgrafik	vektorigrafiikka	vector graphics
resolution	erottuvuus, erotuskyky	resolution
redundanskontroll	ylimäärätarkastus	redundancy check
paritet	pariteetti	parity
jämn paritet	parillinen pariteetti	even parity
udda paritet	pariton pariteetti	odd parity
fil, register, dataregister	tiedosto	file

post	tietue	record
fält	kenttä	field
tecken	merkki	character
bit	bitti	bit

14.3 Datorns uppbyggnad och funktion

styrenhet	ohjausyksikkö	control unit
arimetik-logikenhet	aritmeettis-looginen yksikkö, laskuyksikkö, ALU	arithmetic-logic unit ALU
register, datorregister	rekisteri	register
processor	suoritin, prosessori	processor
primärminne	keskusmuisti	main memory
instruktion	käsky	instruction
prestanda	suorituskyky	performance
klockfrekvens	kellotaajuus	clock frequency
klockcykel	kellojakso	clock cycle
operativsystem	käyttöjärjestelmä	operating system
minnescell	muistisolu	storage cell
ordlängd	sanan pituus	word length
minnesaccess		
byte, bitgrupp	tavu	byte
sekundärminne	apumuisti	auxiliary storage, secondary storage
inre, internt minne	sisäinen muisti	internal storage
yttre, externt minne	ulkoinen muisti	external storage
arbetsminne	työmuisti	working storage
massminne	massamuisti	mass storage
skivminne	levymuisti	disk memory
minnesadress	muistiosoite	memory address
fickminne, cache-minne	välimuisti	cache memory
virtuellt minne	näennäismuisti	virtual storage
programräknare	käskynosoitin	program counter
instruktionsregister	käskyrekisteri	instruction register
instruktionsmängd	käskykanta	instruction set
maskinspråk	konekieli	machine language
bildskärm	näytin, näyttö	display
terminal	pääte	terminal
skrivare	kirjoitin	printer
adressbuss	osoiteväylä	address bus
databuss	tietoväylä	data bus
kontrollbuss	ohjausväylä	control bus
datorarkitektur	tietokonearkkitehtuuri	computer architecture

simulator	simulaattori	simulator
emulator	emulaattori	emulator
kompatibel	yhteensopiva	compatible
mikrodator	mikrotietokone	personal computer, PC, (microcomputer)
		PC, (microcomputer)
arbetsstation	työasema	work station
minidator	minitietokone	minicomputer
generell dator, stordator	yleistietokone	mainframe
superdator	supertietokone	supercomputer
kiselbricka	piisiru	chip

14.4 Kringutrustning

primärminne	keskusmuisti	main memory
sekundärminne	apumuisti	auxiliary memory
åtkomsttid	saantiaika	access time
väntetid, latenstid	odotusaika	latency, waiting time
överföringstid	siirtoaika	transfer time
direktminne	suorasaantimuisti	direct access storage
cykliskt minne	syklinen muisti	cyclic storage
sekventiellt minne	peräkkäismuisti	sequential storage
flyktigt (obeständigt) minne	katoava muisti	volatile storage
beständigt (icke-flyktigt) minne	katoamaton muisti	non-volatile storage
permanent minne	pysyväismuisti	permanent storage
raderbart minne		eraseable storage
halvledarminne	puolijohdemuisti	semiconductor storage
integrerad krets	integroitu piiri	integrated circuit
kiselbricka	piisiru	chip
läshuvud	lukupää	read head
skrivhuvud	kirjoituspää	write head
skiva	levy	disk
magnetskiva	magneettilevy	magnetic disk
skivpacke	levypakka (levykkö)	disk pack
skivsida		disk surface
spår	ura	track
sektor	sektori	sector
cylinder	sylinteri	cylinder
diskett, flexskiva	levyke (”lerppu”, ”korppu”)	diskette, floppy disk
kassettskiva	kasettilevy	
winchesterminne	umpilevy	hard disk, Winchester disk
diskettenhet	levykeasema	diskette drive

skivstation	levyasema	disk drive
magnetband	magneettinauha	magnetic tape
arkivminne	arkistomuisti	archive storage
bandenhet	nauhayksikkö	tape unit
block	jakso	block
post	tietue	record
optisk skiva	optinen levy	optical disk
fördröjningsledningsminne		delay line storage
trumminne	rumpumuisti	drum storage
kärnminne	ydinmuisti	core storage
bubbelminne	kuplamuisti	magnetic bubble storage
tangentbord	näppäimistö	keyboard
råtta, mus	hiiri	mouse
bildskärm	näyttö, näytin	display (unit)
katodstrålerörsskärm	katodisädeputkinäyttö	cathode ray tube display, CRT
bildpunkt	kuva-alkio	pixel
plasmaskärm		plasma panel
flytkristallskärm	nestekidenäyttö	liquid chrystal display
bildrör	kuvaputki	(picture) tube
dataterminal	päätelaite	data terminal equipment
positionerare	paikannin	locator
pekdon		pointing device
mus, råtta	hiiri	mouse
ljuspenna	valokynä	light pen
optisk läsare	optinen lukija	optical reader
digitaliseringsunderlag	digitointialusta	
bildläsare	kuvanlukija	scanner
inmatning via bildskärm	”kosketusnäyttö”	touch-panel screen
styrspak	ohjaussauva	joy stick
styrkula	ohjauskuula	control ball
skrivare	kirjoitin	printer
teckenskrivare	merkkikirjoitin	character printer
radskrivare	rivikirjoitin	line printer
sidskrivare	sivukirjoitin	page printer
anslagsskrivare	iskukirjoitin	impact printer
anslagsfri skrivare	siirtokirjoitin	non-impact printer
matrisskrivare	matriisikirjoitin	matrix printer, dot printer
bläckstråleskrivare	mustesuihkutulostin	ink jet printer
laserskrivare	lasertulostin	laser printer
värmeöverföringsskrivare	lämpökirjoitin	termo transfer printer
elektrostatisk skrivare	sähköstaattinen kirjoitin	electrostatic printer
typhjulsskrivare	kiekkokirjoitin	daisy wheel printer
kurvritare	piirturi	plotter
taligenkänning	puheentunnistus	speech recognition
talsyntes	puhesynteesi	speechsynthesis

grafik, datagrafi, datorgrafi, grafisk databehandling	grafiikka	graphics
ljudutmatning	äänitulostus	
talinmatning	puhesyöttö	
dataglasögon		
datahandske		data glove
virtuell verklighet	virtuaalitodellisuus	virtual reality
robotteknik	robottitekniikka	robotics
hålkort	reikäkortti	punched card
hålremsa	reikänauha	paper tape
magnetbandsinkodare		key-to-tape unit
magnetisk teckenigenkänning	magneettimerkkien tunnistus	magnetic ink character recognition, MICR
optisk märkläsning		optical mark recognition OMR
streckkodsläsare	juovakoodin lukija, ”viivakoodin” lukija	bar-code reader/scanner
inbäddat system	sulautettu järjestelmä	embedded system

14.5 Algoritmer

algoritm	algoritmi	algorithm
numrerade steg		
traditionellt funktionsschema		
strukturerat funktionsschema		
datorprogram	tietokoneohjelma	computer program
sekventiell åtgärd	peräkkäinen toiminto	sequential statement
villkorlig åtgärd	ehdollinen toiminto	conditional statement
upprepad åtgärd	toistuva toiminto	iterative statement
variabel	muuttuja	variable
tilldelning (-soperation)	sijoitus (-operaatio)	assignment (operation)
inmatningoperation, läsoperation	syöttöoperaatio, lukuoperaatio	input (read) operation
utmatningoperation, skrivoperation	tulostusoperaatio, kirjoitusoperaatio	output (write) operation
indatakö, inkö	syöttöjono	input queue
utdatakö, utkö	tulosjono	output queue
kunskap	tietämys	knowledge
hanterligt problem		tractable problem
ohanterligt problem		intractable problem
oavgörbart problem	ratkeamaton ongelma	undecidable problem

14.6 Programspråk

programspråk (programmeringsspråk)	ohjelmointikieli	programming language
maskinspråk	konekieli	machine language
assembleringsspråk, symboliskt maskinspråk	symbolinen konekieli	
högnivåspråk	lausekieli	high level language
applikationsspråk	sovelluskieli	application language
programmeringsparadigm	ohjelmointiparadigma	programming paradigm
imperativt språk	imperatiivinen kieli	imperative language
objektorienterat språk	oliosuuntautunut kieli	object-oriented language
funktionelltspråk	funktionaalinen kieli	functional language
parallellt språk	rinnakkais(ohjelmointi)kieli	parallell language
logikspråk	logiikkakieli	language
abstraktion	abstrahointi	abstraction
underprogram	aliohjelma	subprogram
funktion	funktio	function
procedur	proseduuri	procedure
abstrakt datatyp	abstrakti tietotyyppi	abstract datatype
syntax	syntaksi	syntax
semantik	semantiikka	semantics
kompileringsfel	käännösvirhe	compilation error
exekveringsfel	ajonaikainen virhe	run-time error
logiskt fel	looginen virhe	logical error
maskinberoende	koneriippuvainen	machine dependent
maskinoberoende	koneriippumaton	machine independent
assemblator	kääntäjä	assembler
frågespråk	kyselykieli	query language
generator	kehitin	generator
definierande språk	määrittelevä kieli	defining language

14.7 Programvara

programvara	ohjelmisto	software
systemprogram	varusohjelma	system software
applikationsprogram, tillämpningprogram	sovellusohjelma	application software
operativsystem	käyttöjärjestelmä	operating system
kompilator	lausekielen kääntäjä	compiler
primäranvändare	peruskäyttäjä	end user
filhantering	tiedostokäsittely	file management
katalog	hakemisto	directory

rotkatalog	päähakemisto	root directory
sökväg	hakupolku	search path
arbetskatalog, aktuell katalog	työhakemisto	current directory
hemkatalog	kotihakemisto	home directory
multiprogramkörning	moniajo	multiprocessing
multiprocessor	monikone	multiprocessor
tidsdelning	osituskäyttö	time sharing
satsvis bearbetning	eräajo	batch processing
realtidssystem	tosiaikakäyttöjärjestelmä	real time system
fönsteromgivning	ikkunaympäristö	window envirnment
ikon	ikoni	icon
meny	valikko	menu
interaktivt program	keskusteleva, interaktiivinen ohjelma	conversational, interactive program
redigerare, editor	editori, toimitin	editor
översättare	kääntäjä	translator
virtuellt minne	näennäislmuisti	virtual memory
kompilator	kääntäjä	copmpiler
tolk	tulkki	interpreter
källspråk	alkukieli	source language
målspråk, objektspråk	tuloskieli	object language
källprogram	alkukielinen ohjelma	source program
målprogram, objektprogram	tuloskielinen ohjelma	object program
länkare	yhdistelijä	linker
debugger, avlusare	jäljitinohjelma	debugger
användargränssnitt	käyttöliittymä	user interface
rullgardinsmeny		pulldown menu
blankett	lomake	form

14.8 ADB-historik

räknebord	laskupöytä	counting board
kulram	helmitaulu	abacus
analytisk maskin	analyyttinen kone	analytical engine
bugg	bugi	bug
elektronrör	elektroniputki	vacuum tube
transistor	transistori	transistor
integrerad krets	integroitu piiri	integrated circuit

14.9 Datahantering

datahantering, dataadministration	tiedonhallinta	data management
tillämpningscentrerad dataadministration	sovellutuskohtainen tiedon hallinta	
datacentrerad dataadministration	tietokeskeinen tiedon hallinta	
källdata	alkutiedot	source data
redundans	ylimäärä	redundancy
uppdatering	päivitys	update
datadispersion	tiedon hajaantuminen	data dispersion
databas	tietokanta	database
databassystem	tietokantajärjestelmä	database system
databasadministrationsprogram	tietokannan hallintaohjelmisto	database management system
databashanteringsprogram	tietokannan hallintaohjelmisto	database management system
fysiskt data	fyysinen tieto	physical data
logiskt data	looginen tieto	logical data
fil	tiedosto	file
post	tietu	record
fält	kenttä	field
tecken	merkki	character
byte	tavu	byte
bit	bitti	bit
datadefinitionsspråk	tiedon määrittelykieli	data definition language, DDL
datamanipuleringsspråk	tiedon käsittelykieli	data manipulation language, DML
initialisering	alustus	initialization
sökning	haku	search
bläddring	selailu	browsing
fråga	kysely	query
tillägg	lisäys	insertion
borttagning	poisto	deletion
förändring	muuttaminen	change
sortering	lajittelu	sorting
indexering	indeksointi	indexing
frågespråk	kyselykieli	query language
datamodell	tietomalli	data model
hierarkisk datamodell	hierarkkinen tietomalli	hierarchical model
nätverksmodell	verkkomalli	network model
relationell modell	relaatiomalli	relational model
objektmodell	oliomalli	object model
hierarkisk databas	hierarkkinen tietokanta	hierarchical database

nätverksdatabas	verkkotietokanta	network database
relationell databas	relaatiotietokanta	relational database
objektsdatabas	oliotietokanta	object database
inkapsling	kapselointi	encapsulation
nedärvning	periytyminen	inheritance
objektidentitet	olioidentiteetti	object identity
hypertext, -media	hyperteksti, -media	hypertext, -media
strukturerat dokument	rakenteinen dokumentti, rakenteinen asiakirja	structured document
dataskydd	tietoturva	data security
datasekretess, integritetsskydd	tietosuoja	data protection, privacy protection
integritet	yksityisyys	integrity
känsliga uppgifter	arkaluonteiset tiedot	

14.10 Datakommunikation

datakommunikation	tietoliikenne	data communications
protokoll	protokolla	protocol
simplex	yksisuuntainen	simplex
halvduplex	vuorosuuntainen	half duplex
duplex	kaksisuuntainen	duplex
asynkron överföring	asynkroninen siirto	asynchronous transfer
synkron överföring	synkroninen siirto	synchronous transfer
sekventiell, seriell överföring	peräkkäissiirto	serial transfer
parallell överföring	rinnakkaissiirto	parallel transfer
redundanskontroll	ylimäärätarkastus	redundancy check
paritetskontroll	pariteettivarmistus	parity check
validitetskontroll	kelpoisuusvarmistus	validity check
kontrollsumma	tarkistussumma	check sum
terminal	pääte	terminal
gränssnittsenhet	liitäntälaite	interface unit
datanät	datan siirtoverkko	data network
datornät	tietokoneverkko	computer network
distribuerat system	hajautettu järjestelmä	distributed system
lokalt nät	lähiverkko	local area network, LAN
icke-lokalt nät		wide area network, WAN
bussnät	väyläverkko	bus network
slingnät	rengasverkko	ring network
stjärnnät	tähtiverkko	star network
telefonnät	puhelinverkko	phone network
modem	modeemi	modem
telefax, telekopiering	kaukokopiointi	telefax

telex	telex	telex
teletex	teleteksi	teletex
elektronisk post, epost	sähköposti	electronic mail, e-mail
diskussionsgrupp	uutisryhmä, keskusteluryhmä	news group
webb	webi?, WWW	web
filöverföring	tiedostosiirto	file transfer

14.11 Systemutveckling

systemutveckling, systemarbete	systeemityö	system development
systemanalys	systeemianalyysi	system analysis
systemplanerare, systemerare	systeemisuunnittelija	system designer
datasystem	tietosysteemi	data system
projekt	projekti, hanke	project
förundersökning	esitutkimus	feasibility study
systemplanering	systeemin suunnittelu	system planning, design
implementering, konstruktion	toteutus	implementation
ibruktagande	käyttöönotto	installation
underhåll	ylläpito	maintenance

14.12 Artificiell intelligens, databrott, ergonomi

artificiell intelligens	tekoäly	artificial intelligence
expertsystem	asiantuntijajärjestelmä, taitotuki	expert system
kunskapsbaserat system	tietämysjärjestelmä	knowledge-based system
regelbaserat system	sääntöpohjainen järjestelmä	rule-based system
maskininlärning	koneoppiminen	machine learning
behandling av naturligt språk	luonnollisen kielen käsittely	natural language processing, NLP
mönsterigenkänning	hahmontunnistus	pattern recognition
neuralnät	neuroverkko	neural network
intelligent agent	älykäs agentti	intelligent agent, software agent
faktum	fakta	fact
regel	sääntö	rule
kunskapsingenjör	tietämysinsinööri	knowledge engineer

databrott	atk-rikos	computer crime
trojansk häst	troijalainen hevonen	Troyan horse
tidsbomd	aikapommi	time bomb
salamimetod	”salamimenetelmä”	Salami slice
lönndörr	salaovi	trap door
cracker, systembrytare		cracker
hacker	hakkeri	hacker
datavirus	tietokonevirus	computer virus
virusbekämpningsprogram	virustorjuntaohjelma	antivirus software
ergonomi	ergonomia	ergonomics